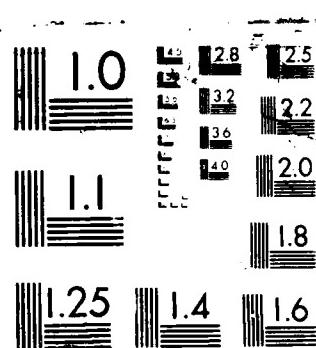


AD-A199 189 JOINT LOGISTICS COMMANDERS' WORKSHOP ON POST-DEPLOYMENT 1/6
SOFTWARE SUPPORT (U) JOINT POLICY COORDINATING GROUP
ON COMPUTER RESOURCE MANAGEMENT JUN 84

UNCLASSIFIED

F/G 12/3 NL



1.0 1.1 1.25 1.4 1.6

2.8
3.2
3.6
4.0

2.5
2.2
2.0
1.8

DTIC FILE COPY

FINAL REPORT OF THE
JOINT LOGISTICS COMMANDERS'
WORKSHOP

ON

DTIC
S TE AUG 11 1988 D
H

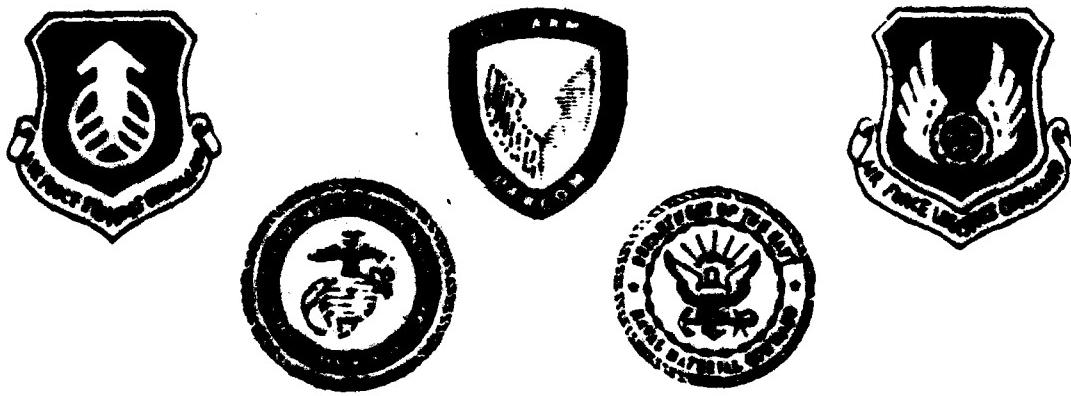
POST DEPLOYMENT SOFTWARE SUPPORT
[PDSS]

FOR

MISSION-CRITICAL COMPUTER SOFTWARE

VOLUME II - WORKSHOP PROCEEDINGS

JUNE 1984



THIS IS NOT AN APPROVED JLC DOCUMENT

DISTRIBUTION STATEMENT A

Approved for public release
Distribution Unlimited

88 810 075

PREFACE

Nowhere can the speed of advancing technology be better exemplified than in the area of computer resources. Our country's weapon systems are inextricably linked to computer hardware and software. Annual Department of Defense expenditures in this technology continue to increase.

> The Joint Logistics Commanders' Joint Policy Coordinating Group on Computer Resource Management (CRM) has been striving since 1977 to achieve sensible triservice policy and standards in the acquisition of computer software in weapon systems where that software is mission critical. The well known Monterey I and Monterey II Workshops have led to Department of Defense software development standards, a tri-service software development policy, and data item descriptions (DIDs) that are expected to be formally implemented in late 1984. Pilot applications of these new documents are currently underway. In addition, a draft policy standard, and DID on software quality is available. The anticipated implementation date for these documents is late 1985.

The CRM group have broadened their area of concern to the entire life cycle of weapon system software by sponsoring an important workshop on software support (often misleadingly called software "maintenance"). This significant workshop, called Orlando I, focused on the issues of modification of software to support mission requirements changes and to improve performance after the development of the initial computer programs and/or after the deployment of the weapon system. This volume outlines the workshop organization, summarizes the speeches of the honored guests of the workshop, and provides the complete reports of the six panels addressing the Post Deployment Software Support (PDSS) issues of government/industry workforce mix, independent verification and validation, cost of ownership, software support environments, the software change process and configuration management. ()

INSTANCED
2

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unrestricted	<input type="checkbox"/>
Justification	
By <i>per letter</i>	
Distribution	
Availability Codes	
Avail and/or	
Dist	Special
A-1	

TABLE OF CONTENTS

	<u>PAGE</u>
Preface	i
Table of Contents	iii
List of Figures	iv
1. Workshop Agenda	1-1
2. Workshop Organization	2-1
3. Guest Speaker Presentations	3-1
3.1 Relationship Between PDSS and Advanced Technology (Dr. Edith Martin)	3-2
3.2 Software (Maj. Gen. Monroe T. Smith)	3-7
4. Workshop Panels	4
A Industry/Government Workforce Mix	4-1-i
Table of Contents	4-1-ii
B IV&V by Support Personnel	4-2-i
Table of Contents	4-2-ii
C Cost of Ownership	4-3-i
Table of Contents	4-3-ii
D Software Support Environment	4-4-i
Table of Contents	4-4-ii
E Change Implementation	4-5-i
Table of Contents	4-5-ii
F Configuration Management	4-6-i
Table of Contents	4-6-ii

LIST OF FIGURES

1.1	Agenda	1-2
1.2	Agenda	1-3
1.3	Agenda	1-4

LIST OF TABLES

2.1	Orlando I Panel Objectives	2-2
2.2	Administrative Organization	2-3
2.3	Panel Co-Chairpersons	2-4

1. WORKSHOP AGENDA

Orlando I began at 1330 hours on 31 October with a welcoming by the general chairman of the conference, Mr. Bill Egan of the Naval Material Command. He introduced the Executive Chairman, Col. John Marciniak and the Executive Committee. (Workshop organization is described in Section 2 of this volume.) The Honorable Dr. Edith Martin, Deputy Under Secretary of Defense for Research and Advanced Technology, was then introduced. She delivered the keynote address which is summarized in Section 3 of these proceedings. Following Dr. Martin's address, the status of Monterey Workshops I and II was reported by LCDR Mike Gehl, chairman of the JLC-CRM/CSM subgroup. The program chairman, Mr. Wayne Sherer, introduced the panel chairmen and the workshop topics. The panel chairmen then summarized to the workshop at large the subjects which they were going to discuss and some of the key issues to be resolved. Individual panel orientations then followed. The agenda for the first day of the workshop is shown in Figure 1.1.

The next three days of the workshop followed an agenda of essentially a similar format. Panel chairpersons met with the executive committee early each morning to report progress, bring forth problems, and receive guidance. Following nearly four hours of panel and subpanel discussion and documenting activity, there was lunch including an interesting and important luncheon speech. Following a two and one half hour afternoon session within panels and subpanels, a joint 60-minute session of all panels was held during which panel conclusions for that day were summarized.

On Wednesday evening a banquet for participants was held. The guest speaker for the banquet was Maj. Gen. Monroe T. Smith, Commander of the Air Force Acquisition Logistics Division and Deputy Chief of Staff for Acquisition Logistics, HQ AFLC.

On the last day of the workshop a morning long joint session was conducted where the week long panel activities were summarized.

The agendas for the last four days of the conference are shown in Figures 1.2 and 1.3.

**Joint Logistics Commanders
Joint Policy Coordinating Group on Computer
Resources Management**

**SOFTWARE WORKSHOP — ORLANDO I
on
Post Development Software Support**

AGENDA

Monday, 31 October 1983

- | | |
|-----------|---|
| 1000-1330 | Registration Check-In (Lobby, Langford Hotel,
Winter Park, FL) |
| 1330-1400 | Welcome |
| 1400-1430 | Keynote Address: Dr. Edith Martin, Deputy Under
Secretary of Defense for Research & Advanced
Technology |
| 1430-1500 | Status Report - Monterey I and II |
| 1500-1600 | Panel Introductions |
| | A. Industry/Government Workforce Mix |
| | B. IV&V by Support Personnel |
| | C. Cost of Ownership |
| | D. Software Support Environment |
| | E. Change Implementation |
| | F. Configuration Management |
| 1600-1615 | Break |
| 1615-1730 | Panel Orientations |
| -1800 | Workshop Reception (hors d'oeuvres, no-host bar,
watershow) |

Figure 1.1

Tuesday, 1 November 1983

0730-0800	Panel Chairpersons/Executive Committee Meeting
0800-0945	Panel Sessions
0945-1000	Coffee Break
1000-1200	Panel Sessions
1200-1330	Luncheon: Guest Speaker: Dr. Robert Mathis, ADA Joint Program Officer
1330-1500	Panel Sessions
1500-1515	Coffee Break
1515-1615	Panel Sessions
1615-1630	Break
1630-1730	Joint Session (Panel Summaries)

Wednesday, 2 November 1983

0730-0800	Panel Chairpersons/Executive Committee Meeting
0800-0945	Panel Sessions
0945-1000	Coffee Break
1000-1200	Panel Sessions
1200-1330	Luncheon: Guest Speaker: Capt. James Van Metre, USN
1330-1500	Panel Sessions
1500-1515	Coffee Break
1515-1615	Panel Sessions
1615-1630	Break
1630-1730	Joint Session (Panel Summaries)
1830-1930	No-Host Bar
1930-	Banquet Guest Speaker: Maj. Gen. Monroe T. Smith US Air Force

Figure 1.2

Thursday, 3 November 1983

0730-0800	Panel Chairpersons/Executive Committee Meeting
0800-0945	Panel Sessions
0945-1000	Coffee Break
1000-1200	Panel Sessions
1200-1330	Luncheon Guest Speaker, Col. James V. Bronson, USMC
1330-1500	Panel Sessions
1500-1515	Coffee Break
1515-1615	Panel Sessions
1615-1630	Break
1630-1730	Joint Session (Panel Summaries)

Friday, 4 November 1983

0730-0800	Panel Chairpersons/Executive Committee Meeting
0800-0945	Joint Session (Panel Summaries)
0945-1000	Coffee Break
1000-1200	Joint Session/Wrap Up

Special Thanks to Wang for its word processing equipment and services during the conference and to IBM Corporation for use of copying machine.

Figure 1.3

2. WORKSHOP ORGANIZATION

As indicated in Section 1, the Orlando I Post Deployment Software Support workshop was organized into six separately functioning panels. The objective of each panel is shown in Table 2.1. Each panel was co-chaired by government and industry chairpersons. Table 2.2 presents the names, affiliations, addresses and phone numbers of these chairpersons. The chairpersons were responsible for the daily panel summaries and the panel reports which were assembled in rough draft form prior to the end of the workshop. During the months since the workshop the panel reports have been reviewed by each panel member, corrections have been made and the final panel reports have been prepared. These reports are in Section 4 of this volume.

In addition to the technical organization of the conference, there was considerable planning and other administrative activity involved. Committee chairpersons and managers are listed in Table 2.3.

ORLANDO I

Mission-Critical Post Deployment Software Support (PDSS) Workshop

PANEL OBJECTIVES

Panel A - INDUSTRY/GOVERNMENT WORKFORCE MIX

Develop policy recommendations for cost-effective staffing of software support agencies using appropriate mixes of government and industry personnel.

Panel B - INDEPENDENT VERIFICATION AND VALIDATION (IV&V)

Determine when and how much IV&V should be used in software development and during Post Deployment Software Support (PDSS).

Panel C - COST OF OWNERSHIP

Clarify the basis of large projected costs of future software development and support while identifying approaches to reducing software cost.

Panel D - SOFTWARE SUPPORT ENVIRONMENT

Discuss the requirements for establishing an effective, generic post deployment software support environment establishing feasibility, advantages and disadvantages.

Panel E - THE SOFTWARE CHANGE PROCESS

Develop the framework for a joint services PDSS "Change Policy Manual."

Panel F - CONFIGURATION MANAGEMENT

Determine a common definition and scope of "software configuration management" which is suitable to be promulgated by the JLC.

Table 2.1

ADMINISTRATIVE ORGANIZATION

Executive Chairman:

Colonel John Marciniak, USAF

Executive Committee:

Capt. Dave Boslaugh, USN
Col. Ken Nidiffer, USAF
Lt. Col. James Harrington, USAF
Col. H. R. Archibald, US Army
Maj. K. R. Ptack, USMC

General Chairman:

Mr. Bill Egan, Naval Air Systems Command

Program Chairman:

Mr. Wayne Sherer, U.S. Army Armament Munitions & Chemical Command

Facilities Chairman:

Capt. Tom Smith, US Marine Corps

Publications Chairman:

Maj. Ed Stevens, HQ AFSC/ALR
Capt. Lee Cooper, HQ AFSC/ALR

Special Arrangements:

Mr. Mert Batchelder, HQDARCOM

Protocol Officer:

Lt. Sunny Riley, HQAFLC/MMEC

Administration/Business Manager:

Ms. Roxy McCarter, HQNAVMAT

NTEC Liaison:

Mr. Frank Jamison, Naval Training Equipment Center

Workshop Manager:

Ms. Michele Foley, P/M Group

Planning Support:

Ms. Dreama Fumia, Veda, Inc.

Treasurer:

Mr. Daniel Kvenvold

Table 2.2

PANEL CO-CHAIRPERSONS

Panel A - Government/Industry Workforce Mix

Lt Col Frank J. Sisti
HQ DA (DAMO-C4L) Pentagon
Washington, DC 20380
(202) 697-4539
A/V 227-4539

Mr. R. Dean Hartwick
Logicon, Inc.
255 West 5th St.
San Pedro, CA 90731
(213) 831-0611

Panel B - Independent Verification and Validation (IV&V)

Cdr D. (Dave) Southworth
HQ, Naval Material Command (MAT 08Y)
Washington, DC 20360
(202) 692-3966

Mr. John W. Sapp
Software A&E, Inc.
1401 Wilson Blvd., Suite 1220
Arlington, VA 22209
(703) 276-7910

Panel C - Cost of Ownership

Lt Col James Riley
HQ AFSC/DLA
Andrews AFB
Washington, DC 20334
(301) 981-2482

Mr. G. (Gene) Sievert
Teledyne-Brown Engineering
300 Sparkman Dr.
Huntsville, AL 35807
(205) 532-1500

Panel D - Software Support Environment

Mr. Jim Hess
HQ DARCOM/DRCDE-SB
5001 Eisenhower Avenue
Alexandria, VA 22333
(202) 274-9318
A/V 284-9318

Mr. Jerry Raveling
Sperry Corporation
Computer Systems, M.S. U1E13
P.O. Box 43525
St. Paul, MN 55164
(612) 456-3545

Panel E - The Software Change Process

Mr. Joe Black
WR-ALC/MMRR
Robins AFB, GA 31098
(912) 926-5948

Mr. Jack Cooper
CACI, Inc.
Federal Penthouse
1700 N. Moore St.
Arlington, VA 22209
(703) 276-2826

Panel F - Configuration Management

Mr. C. (Cal) Showalter
Naval Air Systems Command
(AIR-543C)
Room 620, JP-2
Washington, DC 20361
(202) 746-0650

Ms. Antonia D. Schuman (Toni)
TRW Systems Group
1 Space Park, Bldg. 134
Room 6079
Redondo Beach, CA 90278
(213) 217-4079

Table 2.3

3. GUEST SPEAKER PRESENTATIONS



The keynote address was delivered by Dr. Edith W. Martin. Prior to her appointment as Deputy Under Secretary of Defense for Research and Advanced Technology, Dr. Martin was an executive with Control Data Corporation and director of Government Systems, Atlanta operations. Previously, Dr. Martin was director of the Computer Science and Technology Laboratory at Georgia Tech's Engineering Experiment Station. In both of these positions, she was involved in Defense-related research and development activities. Recipient of numerous awards for her leadership and contributions to the defense/industrial community, she has also served as a member of the Defense Science Board Special Task Force on Embedded Computer Resources and as chairman of a large number of technical review committees.

Dr. Martin's address was entitled "The Relationship Between Post Deployment Software Support and Advanced Technology." The script for this address is presented in Section 3.1 of this volume.



The banquet speaker was Maj. Gen. Monroe T. Smith, who is Commander, Air Force Acquisition Logistics Division, and Deputy Chief of Staff for Acquisition Logistics, Headquarters Air Force Logistics Command, Wright-Patterson Air Force Base, Ohio. Previously the general was Deputy Chief of Staff/Maintenance at HQ AFLC, Wright-Patterson Air Force Base. Prior to that he was commander of the Defense Contract Administration Services Region in Los Angeles, Director of Materiel Management for the Sacramento Air Logistics Center, and Director for Plans and Industrial Resources at HQ AFLC at Wright-Patterson. Prior to the above assignments, General Smith was a research associate for the Fletcher School of Law and Diplomacy and functioned as Chief of Executive Services and Maintenance Staff Officer in the Aircraft Systems Division, Office of the Deputy Chief of Staff for Systems and Logistics, HQ USAF in Washington, D.C.

General Smith's banquet address is entitled "Software" and is presented

in Section 3.2 of this volume.

The luncheon speakers were Dr. Robert Mathis, Technical Director of the Ada Joint Program Office (AJPO) in the office of the Deputy Under Secretary of Defense for Research and Advanced Technology; Captain James Van Metre, U.S. Navy, Project Manager for the Submarine Advanced Combat System; and Colonel James V. Bronson, Commanding Officer, Marine Corps Tactical Systems Support Activity.

3.1 RELATIONSHIP BETWEEN PDSS AND ADVANCED TECHNOLOGY (Dr. Edith Martin)

THE RELATIONSHIP BETWEEN PDSS AND ADVANCED TECHNOLOGY

It's a pleasure for me to address you today for several reasons. First, because I am the DOD principal responsible for MCCR policy--the implementation of the Warner Amendment to the Brooks Act. Clearly, PDSS and the way that we manage our mission-critical software are very important to me. Second, because of my responsibility for the technology base in computer systems and software--including Ada and STARS. Third, because it brings back memories of an important workshop that I co-chaired in Orlando several years back on Ada and Nebula--before either of these important standards was solid. And fourth, because, unlike, for example, addressing an audience on fuels or lubricants, where I can't find anybody I know, here I see many old friends and familiar faces.

Today, I would like to talk about the relationship between PDSS and advanced technology. Both are worlds of change--sometimes very rapid change. PDSS presents an important challenge to technology: providing the capability to create software and systems that are truly dynamic--adaptable. Technology, on the other hand, challenges PDSS: accept and integrate advances into existing PDSS functions and structures as quickly as possible.

REQUIREMENTS FOR EFFECTIVE SOFTWARE

All of us I'm sure will readily admit that we are not happy with the state of affairs of the software life cycle. I think we should start by stating what are we really looking for. Then we can keep asking the question--What are we doing to satisfy these requirements? Are we making any progress?

I see that we have requirements in two categories--technical and management. Technical--the desired properties of software, and second, the desired management situation. I will not focus at this point on differences in life cycle emphasis--all of the phases are equally important.

If you tell me that PDSS deserves more emphasis because it consumes 70% of the life cycle costs, then I will tell you--the way we do that 30% or whatever of development has tremendous leverage on the rest of the life cycle--both positive and negative. If you tell me that development is where it's at, then I'll say that the whole purpose of development is to provide the evolutionary flexibility we need in PDSS. You can almost guarantee that if development is done well, then PDSS will be facilitated, and if development is messy then PDSS is going to be a problem. Too often software development and PDSS are treated as separate universes. Those that are primarily concerned with PDSS see the world from that vantage point and those that are involved only with development frequently have a similar narrow focus. We should view these sometimes separate activities as a continuum.

In reality, support is done during development and development is done during support. It's only a difference of emphasis. The primary issues that we face in software exist in both of these worlds.

TECHNICAL REQUIREMENTS

Software should be responsive to the system specifications driving it. It should also have sufficient inherent adaptability to facilitate correction of errors as well as redesign as needs evolve due to changes in the threat or changes in hardware. Software should be reliable--meaning it shouldn't cause system failures. It should be general enough so that it could be used in other applications or in future versions of the current application. Finally, it should be transportable to a new computer, whether or not the same architecture is used.

MANAGEMENT REQUIREMENTS

Looking at software from a management perspective, we want effective means to control schedule and costs. We need an adequate supply of software professionals with the appropriate level of training and expertise to carry out the software function. We must be able to accomplish development in the most appropriate way and do the same for PDSS--and we must have policies and mechanisms for transitioning between the two in the most efficient manner.

The approaches we set up in PDSS operations to correct problems and to evolve software must be highly responsive and cost effective. Perhaps most important, our management structures should not allow us to lose sight of the "total system." We must not forget that we are engineering systems of which the software is a part. When we change software we are, in effect, making engineering changes to the entire system and while the software change may be easy, the consequences may be catastrophic. We should be careful not to separate the software too far from the remainder of the system. And we should never allow software and system management to be at the same level. When we do, we really have nobody in charge.

EXPONENTIAL GROWTH OF SOFTWARE

The growth of software cost and demand has been and continues to be exponential. Because of this, complexity of the type of software in DOD desire has increased to the point where our practices are now incapable of producing software without a litany of problems.

CURRENT SITUATION

The main problems we have been experiencing in software are listed here. Software has, all too frequently, been the cause of major slips in weapon system schedules. Software costs are now increasing out of proportion to other system costs. The EIA has predicted that annual mission critical software costs can be expected to be 10% of the defense budget by 1990 if current productivity levels continue. We are all familiar with cases where software errors have caused system failures even though the hardware was working perfectly. Software is difficult to modify and finds little reuse elsewhere. Last but not least, we don't have enough software people. It's been estimated that the need for software professionals is increasing at the rate of 12% annually, yet the supply is increasing at only 4%

per year. We assume that productivity would normally increase by 4% annually. This leaves a yearly shortfall of 4%. If nothing changes the current trend, this number will grow to almost one million by 1990.

NO SIMPLE SOLUTION

There is no single or obvious solution to the "software problem." We expect that we will have to do all of these things at a minimum.

THE SOFTWARE ENVIRONMENT

The software environment is where many things come into focus. If we view development and PDSS as a continuum, which I think we should, even though there may be a "handover" in most cases, then that which is required during initial development is also needed for PDSS.

The PDSS environment also includes the interface to operational systems so it, in effect, is a superset of the development environment.

Well, most of the software is developed by contractors. The government functions primarily in a monitoring capacity. During PDSS, the government has more of an operational role.

Should the environments be different or the same? The answer to this is not straightforward. If we say these environments should be the same, then this type of reasoning would lead us to conclude that we should have standard environments in each service. Extension of this reasoning would lead to a single standard environment in both government and the defense industry. This would be a mistake. The field is moving too quickly and we should not take any action to stifle this progress toward the solution of our problems. The worst thing we could do would be to cast our first successful Ada environment in concrete. What we should seek is commonality and standardization where it makes sense to have them--and to evolve standards in the normal fashion--and to allow rapid technological growth in those areas where standardization does not make sense.

COMMONALITY IN THE SOFTWARE ENVIRONMENT

The benefits achievable through the use of a common high order language are well known. This is standardization at one level of interface. The interfaces of the kernel Ada programming support environment, or KAPSE, is another level. This level will facilitate the movement of software tools and data between APSE's without conversion or reprogramming. This will extend the advantages of Ada into the environment. Work in this area is being carried out under the AEGIS of the AJPO by the KAPSE interface team, or kit, which is government, and the KAPSE interface team from industry and academia, or KITIA. The end product is the common APSE interface set or CAIS, version 1.0 of which is now not out for public review.

My message then is that we should seek to standardize at levels which will benefit us and avoid standardizing at levels which will impede our progress.

ADA COVER

Now I'd like to give you a brief status update on the Ada Program.

THE ADA PROGRAM

I'm sure that most of you know that Ada is now both an ANSI and a MIL standard. Technical features of the language such as exception handling and concurrent processing directly address the needs of mission critical and real time systems and permit us to avoid most embedded code.

Congressional support for Ada has been very strong to the point where Congress wants us to accelerate the entire effort. This, along with our conference on the maturity of the language, has motivated our recent policy which contains mandatory dates for the introduction of Ada.

Pending issue of DOD Directive 3405.XX, which will replace DOD instruction 5000.31, the dates that have been established are 1 January 1984 for systems entering advanced development and 1 July 1985 for systems entering full-scale engineering development. A compiler validation capability now exists and has been used to certify the NYU interpreter and both the ROLM/Data General and the Western Digital Ada Compilers.

THE STARS PROGRAM

The STARS program will have a tremendous positive effect on the entire software life cycle. Its goal is to reduce the labor-intensiveness of the software process and thereby reduce costs and improve quality. Complexity will be reduced and adaptability enhanced. We seek order-of-magnitude improvements in both productivity and reliability by 1990. All of this is in recognition of the fact that software technology has become critical to U.S. world leadership.

STARS will be a specially managed joint service and OSD program for seven years. We now have a STARS joint program office alongside the Ada joint program office in our new Directorate for Computer Software and Systems.

STARS SUBGOALS

In order to satisfy this goal, we will have to make significant improvements in the human resource area, in the technical practices we employ to develop and support software, in the computer systems within which such software will execute, and in our acquisition and project management practices.

STARS TASK AREAS

Shown here are the task areas identified by the STARS task force within which STARS work will be done.

WATERFALL

This chart is intended to indicate the relative scopes of Ada and STARS. Ada's primary impact will be on those phases of the system life cycle shown in the shaded area. Ada is a tool primarily for program or module development. Since its level is higher than that of most other languages, it will also be valuable in detailed design, for example, as a program design language. It can also be expected to have a positive impact on the software integration phase. STARS, however, encompasses all of the phases of the life cycle within its scope.

STARS EMPHASIS

The most critical product of the STARS program is the "integrated" software environment, a highly automated computer system containing a wide range of advanced software tools. This product--or set of products--will function at the heart of the software development and post deployment software support processes for every defense/weapons system that contains a computer. Its purpose is to serve as the principal mechanism for engineering the military software. Automation of many of the software functions promises to increase productivity and reduce the probability of errors.

Today, we do not have "integrated" software environments. Instead, we use primitive systems that are largely manual and highly error prone. The automated tools that do exist support only a small portion of the life cycle. Extension of these environments into cohesive systems of software tools employing advanced methodologies and techniques that collectively automate a high percentage of the software development and support processes will be a key element of the solution of the software crisis.

STATUS OF THE STARS PROGRAM

The STARS program is now being staffed both within OSD and the services and defense agencies. STARS was kicked off in FY 83 through reprogramming and I expect many in this audience are now involved in executing various portions of the program. For FY 84, we survived authorization but are not all the way through appropriations. Presently, we are developing a detailed implementation plan that will build on the STARS strategy document of this past March.

ENDORSEMENTS OF SUPPORT

STARS has received strong statements of support from the organizations shown here. We also have the support of Mr. Weinberger, Mr. Thayer, and Dr. Delauer. Many industrial and academic organizations have also expressed strong support for the program.

PRODUCTS

If one is to be successful in an undertaking, one must have both top-level goals and detailed expectations. At this point, and I caution that we are still quite early in the program, we have delineated this set of products as output. It is general now but it will become much sharper during the next year.

RESULTS

Similarly, one should be able to express a state of well-being that one would hope to be in when the program is declared successful. Will we accomplish all of this? We believe so. As in the case of products, as time goes on we will focus each of these better. It is a difficult undertaking. We welcome your advice and assistance.

NO SCREEN

Back to Orlando I--the work you do here on workforce issues, IV&V, the cost of

discipleship, the software environment, change implementation, and configuration management will form the basis for policies and software practices in the future. You have all been invited based on your known contributions to either software technology or software management. You have undertaken a significant challenge. I too am strongly committed to making things very much better in software. Therefore, I wholeheartedly endorse what you are about to do and I wish you every success.

3.2 SOFTWARE (Maj. Gen. Monroe T. Smith)

SOFTWARE

Good evening ladies and gentlemen. I am supposed to say how pleased I am to have this opportunity to talk with you about post development software support of mission critical computer resources. Let me tell you how pleased I am. Yesterday morning I was up early at 0430 to catch a plane for Washington, DC...spent the last two days discussing software and reliability, maintainability and availability of the advanced tactical fighter with the scientific advisory board...folks who are basically concerned with how far, how high, and how fast...got out of there just in time to catch an aircraft to come down here...and will catch an aircraft 2300 tonight for California to a Commanders' Conference...it's not a pleasure to be here...it's a minor miracle! I would have liked to spend some time here...I was raised about 60 miles from here...would have liked to revisit some of the strawberry fields I worked in as a boy.

If you are here from one of the other services, you may or may not recognize the unique position I hold within the Air Force. My job as the Commander of the Air Force Acquisition Logistics Center, reporting both to General Mullins, AFLC/CC, and General Marsh, AFSC/CC, allows my organization to form a bridge between system development and long term support. The logisticians I have, both AFLC and AFSC resources, are embedded in the SPO doing the logistics job the SPO needs done, and that includes ensuring that mission critical computer software does its intended job...and that once acquired...it is supportable. A rare opportunity to excel.

Every time I use the word opportunity, I am reminded of a story about the devout Christian who lived along a flood-prone area. When asked what he would do if a flood came--he was always quick to point out he didn't have to worry about that...the Lord would take care of him. Well, the inevitable flood came...and he was sitting on his roof as the water swirled around his house...but he was the epitome of calm...almost serene...he knew the Lord would take care of him. About then a rescue boat came up...the devout man said, "Save the others...the Lord would provide."...so the boat went on and saved others in the neighborhood. The river continued to rise...and a helicopter appeared...threw down the lifting device...but he shunned it...telling the crew to "save the others...the Lord would provide." The river continued to rise...and took the devout man under...and he drowned! Standing before the Lord, the devout man cried..."My Lord why did you forsake me...I kept your commandments...I lived my life for you...Why did you forsake me." The Lord shook his head...forsake you...first I sent a boat...then I scrounged up a helicopter...man, I just couldn't get you off that rooftop! I guess the moral of the story is...observe carefully the things that come your way...one of them may be the opportunity of a lifetime...and you may not get another...

My new command is sort of like that...I always wanted to work for a 4-star

general...I never dreamed of working for two 4-star generals at the same time...maybe this is the opportunity of a lifetime.

We have gone a long way in post development software support. As you well know, computers and computer software are an integral part of almost all of our modern defense systems. The successful operation of these systems hinges upon the performance and flexibility of the computer. In a very real sense, software is the "glue" that holds our weapon systems together. Software that does not function correctly could result in the failure of a weapon system to successfully complete its assigned wartime task. Software that cannot keep up with the changes around us could also result in the failure of a weapon system to meet new objectives. Software that is so unique, so different, or so difficult to maintain that it requires excessive resources to sustain...is also a monumental failure.

Our advancing technology and modern system design practices are resulting in flexible weapon system designs—flexible in the sense that by just changing the software, we can effectively counter new threats or add new combat capability to a weapon system. These entities: the weapon system designer...the system and subsystem designer...the computer designer...and the software logistician must work together. They must...one...ensure support is considered in the initial design; two...ensure proper capabilities are in place to provide post development software support; and three...provide post development software support to the user. Are we doing that to the best of our ability? Are we attacking problems head on...forgetting interservice and intraservice rivalries? Are we getting rid of the "not invented here" syndrome? Let's see.

When we start categorizing the various kinds of mission critical computer resources, we in the Air Force Acquisition Logistics Center find it useful to separate the computer resources into five categories (You may categorize them differently...the principle will remain the same)...Avionics; command, control, communications, and intelligence (C3I); automatic test equipment; electronic warfare; and automatic training devices.

Avionics software which equates to the other service operations software is our first category. This category includes such things as weapons delivery computers on our fighter aircraft. Here the post development software support must be responsive to the users' unique changing roles, missions, tactics, threats, etc.

How smart are we being here? A fully integrated digital weapon system must have an integrated software support center...a center to do diagnostics and tests...to integrate changes into the software whether a result of imbedded "bugs" or changes in requirements. Have we truly thought through the initial and long-term costs of such a facility? Does the contractor build one at his facility to do the initial work and later we build another at an organic site? Are we so "loose" with our individual systems architecture we cannot describe concrete interfaces which might allow for less elaborate software support centers?

A second category of mission critical computer resources addresses those computer systems embedded in our command C3I systems. The recent employment of the E-3A sentry to monitor Libran operations is an example of our reliance on a C3I system. A unique problem here is the ability of the post development software support capability to maintain system interoperability among the various C3I systems being supported. What I'm saying here is have we structured a system so complex...and

so large...with so many players having the capability to input changes...I'm wondering if we have true control over the software. Have we truly put aside all the us--them...whoever us and them are...to get a handle on systems such as that? Have we insisted on designs that can be compartmentalized when they're needed...and if not...insisted upon a real software support manager who has real authority? This will become more and more difficult as the infrastructure of various C3I systems becomes more complex.

Automatic test equipment is probably the least glamorous of all the embedded computer systems categories...and may be our biggest problem. Any of you out in the audience who had anything to do with bringing the F-15/F-16 intermediate test stations into being ought to be ashamed of yourselves. To have fighter systems that must deploy and fight all over the world...and be tied to from 3 to 6 C-141 loads of test equipment at squadron level is unacceptable. If we can't have avionics that lasts for 2000 hours...that has finite built-in test...that has graceful degradation when it does begin to fail...and then a suitcase tester for flight line work...then you don't have true combat capability. If we don't get on with the development of generic testers...common across the services...then we deserve the "forced marriages" that OSD gives us from time to time when we cannot agree on courses of action.

The electronic warfare category of embedded computer systems is an extremely volatile area. I believe we have arrived at the point where, unless we can effectively perform the post development software support mission for electronic warfare systems, we will be at an extreme disadvantage in a combat environment. In this category, probably more so than the other, the intelligence community, the software support community, and the man in the cockpit must work together to assure mission accomplishment. If we are not careful, the bureaucrats will institutionalize a business as usual effort in the arena that insures we do not react in a timely manner. This area, more than any other, needs innovative thinking... amongst the services...to insure we can incorporate changes to threat changes.

The last of our five categories is the training device category. We are heavily reliant on automated training devices for training our aircrews, i.e., our sophisticated flight simulator, and for training our maintenance people, i.e., maintenance trainers, a couple of the unique problems that have to be addressed in this category and we haven't done a good job here: how you maintain a high degree of similarity between the prime systems and the training device? And how do you maintain control (or do you maintain control) of the constantly changing configuration of the general purpose computer systems that dominate these training device systems? Some of our trainers are so far behind, the training is worse than none at all...because it does not reflect what really happens in the system.

Basically, the post development software support capability for each of our five categories can be stated in two common terms. First of all, the post development software support capability must be able to fix bugs and/or correct design deficiencies. Secondly, it must be able to perform design enhancements (req - design - code - test). There are problems common across all categories but there are also some problems that are unique to each category. Some of the common problems include some of the issues that this group addressed at Monterey I and II. For example, appropriate design considerations, use of standards, and adequate documentation.

I understand in your previous two sessions, Monterey I and Monterey II, that you concentrated on acquisition of weapon systems computer resources--that you developed an acquisition policy which will be implemented through joint service regulation and associated standards. I'm not convinced this group really believes in standards. Do we all believe in 1750? In 1553? Or any number of standards in existence for years? If so, why do we have weapon systems coming off the drawing boards with waiver after waiver...each to require a unique set of talent and equipment to maintain. Without standards, it will be impossible to ensure that appropriate post development software support capabilities are acquired and installed in sufficient time to meet the most demanding weapon system requirements.

I am well aware that to accomplish acquisition of critical post development software support capabilities, a thorough understanding of unique software support requirements is essential. This is where you folks come in--in your deliberations here in Orlando, I would expect that you will lay the foundation for the "logistics support" for future embedded software.

For those of us in the Department of Defense, if our 4-star bosses were here, I believe they would urge us and our industry counterparts to proceed in the same spirit of the first two sessions. That spirit being one of trying as best we can to ignore individual service parochialism and internal service command parochialism and the tendency to think our way is the best way.

I'm convinced that we face common problems and we can capitalize on each other's knowledge in this forum and this country will come out ahead. We don't have the time or the resources to continue to go it alone while giving the appearance of cooperation.

The structure of your panels indicates to me that you know some of the key issues and problems that have to be addressed if we are to provide the kind of post development software support that our operational forces deserve.

For those of you working on the Configuration Management Panel, you should remember that it hasn't been that many years ago that we didn't know how to manage software. I would hope in your deliberations on configuration management for post development software support that you would take a fresh look at the four functions of configuration management. Look at how we, the services, and our industry suppliers, can most effectively accomplish the configuration management job.

Configuration identification, control, status accounting, and audit are the configuration management jobs that have to be done (in spite of the engineers) to ensure a successful program. Configuration management maintains interoperability, provides a smooth transition from developer to supporter, guarantees the ability to upgrade systems, and avoids high support costs of lost configurations. Your effort in this area is to consider technologies and tools which will facilitate these four tasks.

In your working on the Software Support Environment Panel in some respects, you, more than the other panels, are more closely tied to the acquisition initiatives covered at Monterey I and II and other acquisition related initiatives that have begun in the last 2 years. You will have to factor into your deliberations the decisions related to Ada. The Ada decision is a common thread that has now woven

all the DOD together. In some areas, it has become practical for the services to pursue standardization in different directions. However, this common thread we now have may allow you to address the software support environment in a new light. For example, some quote--economics of scale--unquote similar to ones used in the hardware support may be appropriate in the Ada software support environment. Remember we should maintain our uniqueness if dictated by our operational requirements, but we should also pool our resources and rely on one another where it makes sense.

The Change Implementation Panel has some real challenges that need to be tackled at this conference. When I spoke earlier about the electronic warfare category, I commented on the EW's need for rapid changes in direct support of operations. I have also briefly talked about the need for configuration management. I see the requirement for rapid changes spreading into areas other than electronic warfare. It is a fine line that has to be walked between this rapid change requirement and the configuration management tasks. I believe that if we can design our systems for rapid changes and properly structure our configuration management schemes, we can do both jobs faster and better. The change implementation panel must keep in mind that for operational readiness, change turnaround is the key to successful mission accomplishment, and that configuration management is the chain that keeps that key from being lost.

In the Cost of Ownership Panel, you really have your job cut out for you because we don't really know how to estimate in this area. Remember to put cost in its proper perspective. In order to make decisions on where, by whom, and on what post development software support should be made, managers need to get a handle on what our current costs are and on what the future cost of ownership will be. I believe your discussions will probably lead you into areas such as interservice support and contractor or organic support. You should also discuss the factors and tools related to cost accounting: applying them to software support and how much software support really costs us. The cost of ownership panel should consider cost as a variable in post development software support, not as the solution. I want you to remember that the most cost effective capability does us no good if it doesn't do the job. Similarly, if we have to open the door to Fort Knox, that's not a viable solution either.

The Industry Government Workforce Mix Panel has the unenviable job of addressing the people problem. You need to consider how we allocate the scarce resource of software engineering talent within the government. We in DOD have to acquire and maintain the proper staffing to do the post development software support job. We have to consider the competition (especially from video games) and our ability to recruit and retain the required talent to do the job. The government and government related industries have a challenge acquiring and retaining the software engineers--and let's not kid ourselves, this gets tougher every day. We should make it our goal and the challenge to the industry government workforce mix panel to discuss and recommend an environment where we can not only make the most of this scarce talent, but also, make the career of software engineering in DOD and DOD related industry both desirable and gratifying. Have we gotten so many little pockets of experts in the major commands doing their thing?

Those of you on the Independent Verification and Validation (IV&V) Panel will be discussing a fairly well-proven acquisition concept. IV&V by support personnel has worked well on several DOD acquisitions. For post development software support, the question is, "Who does the IV&V if the support personnel are doing

the changes." We should consider the organizational independence and other factors normally associated with IV&V activities. Questions that need to be answered include: Should one service perform IV&V of hardware and another service review software changes? Should separate organizations within the service perform the IV&V functions? And how does the service or organization performing IV&V get funded for their efforts? We should also consider maintaining consistent sets of definitions about why we do IV&V during acquisition and why we need it in post development software support environment. The IV&V panel should keep in mind that although IV&V is being used on several systems today, the administration and implementation of the IV&V processes are still in their infancy. Guidelines are in existence for the application of IV&V, but the details to implement those guidelines need to be worked out and should be the goal of the IV&V by support personnel panel.

The six panels established here at Orlando I have some unique post development challenges that must be addressed. Many of the challenges have common threads from panel to panel. No one panel will be able to operate in a vacuum, so communication among panels is essential. We need your best efforts operating as individual panels but always keeping in mind that each panel is a part of the total conference. I'm confident your efforts will give us the edge we will need in the post development software support environment.

Let me leave you with a conversation I overheard at a party...of engineers--all kinds...and a software engineer was berating an architectural engineer about the cost of housing--and how they have let the costs get out of hand...very pompously the software engineer related how if the housing industry had improved or the cost of storing a bit of memory on software the house would cost less than 10.00 today. After thinking about that for a while the AE replied that if they designed houses like software engineers designed systems--one wookpecker could destroy civilization.

Thank you very much for asking me to speak and for your attention.

4. WORKSHOP PANELS

ORLANDO I
FINAL REPORT
PANEL A
INDUSTRY/GOVERNMENT WORKFORCE MIX

Co-Chairman: Frank J. Sisti, Lt Col
HQ DA (DAMO-C4L) Pentagon
Washington, DC 20380
(202) 697-4539
A/V 227-4539

Co-Chairman: Mr. R. Dean Hartwick
Logicon, Inc.
255 West 5th St.
San Pedro, CA 90731
(213) 831-0611

TABLE OF CONTENTS

	Page
4.1 Panel A - Industry/Government Workforce Mix	4-1-1
4.1.1 Objective	4-1-1
4.1.2 Scope	4-1-1
4.1.3 Approach	4-1-1
4.1.4 Discussion	4-1-4
4.1.4.1 PDSS Background	4-1-4
4.1.4.2 Definition of PDSS	4-1-5
4.1.4.3 Description of PDSS Environment	4-1-7
4.1.4.4 Workforce Utilization Assessment	4-1-8
4.1.4.5 Taxonomy of Mission Critical Computer Resources . .	4-1-9
4.1.4.6 Software Support Activity Drivers	4-1-10
4.1.4.7 Conclusions	4-1-12
4.1.5 Recommendations	4-1-16
Appendix A - Panel A Participants	4-1-A-1
Appendix B - Bibliography	4-1-B-1
Appendix C - Letter to Panel A Participants	4-1-C-1
Appendix D - Panel A Briefings	4-1-D-1
Appendix E - Workforce Attribute Analysis.	4-1-E-1
Appendix F - Taxonomy of Mission Critical Resource Software	4-1-F-1
Appendix G - Attributes Driving Workforce Selection	4-1-G-1

LIST OF FIGURES

	<u>Page</u>
4.1-1 Interrelationship of Government/Industry Parameters	4-1-3

LIST OF TABLES

4.1-1 PDSS Transition Points	4-1-6
4.1-2 PDSS Driver Attributes	4-1-6
4.1-3 ASPJ Model Analysis	4-1-13
4.1-4 B-52 Weapon System Trainer Model Analysis	4-1-14
4.1-5 AN/TSQ-73 Model Analysis	4-1-15

4.1 PANEL A - INDUSTRY/GOVERNMENT WORKFORCE MIX

4.1.1 Objective

For the Joint Logistics Commanders Joint Policy Coordinating Group (JLC-JPCG) on Computer Resource Management, develop recommendations on the policy that should be followed to cost-effectively staff software support agencies using a mix of government and industry personnel.

4.1.2 Scope

Currently each service has taken its own approach to determining personnel mixes for post-deployment software support agencies. Although many of these approaches use arbitrary percentages as guidelines, there do not seem to be any widely accepted criteria for arriving at the appropriate mix. The application of criteria which have a substantial technical basis will aid the software support activity and improve the utility of automated data processing for mission-critical computer resources.

4.1.3 Approach

The panel consisted of the 22 members named in Appendix A and Dan Kvenvold of the Computer Software Management subgroup. The panel had been informed of panel issues by a letter from the co-chairmen, and members had been requested to prepare position papers on aspects of the charter issue (Appendix C). Twenty position papers were prepared and used by the panel. Following the workshop opening session, the panel convened and discussed organization, schedule, and the panel charter.

The following day, the panel heard presentations from the different services during which the post-deployment software support (PDSS) environments were discussed for mission-critical computer resource applications. These presentations emphasized factors that currently drive the services' allocation of personnel. The panel then divided its broader issue into the following questions:

1. What is the definition of PDSS?
2. What are problems in the PDSS environment that arise from current funding/procurement practices?
3. What are the qualitative distinctions between classes of personnel who support PDSS?
4. What is the taxonomy of DOD systems, and how do different taxonomic elements impact personnel assignment?
5. What are the attributes of PDSS that drive the selection of personnel?

The panel conceived the selection of government/industry workforce mix to be an application of a model represented by the three-dimensional drawing in Figure 4.1-1. For each system being considered, elements within the three-dimensional space must be analyzed against the attributes along each dimension (as the attributes are either generally true or not true for that specific application/environment). From a consideration of this information, a trade can be conducted to select the optimal workforce composition. The analysis of these three axes in effect constitutes answers to questions 3, 4, and 5.

Three subpanels were convened to analyze these questions. Each panel selected a chairman and a scribe. These subpanels and the questions they considered were:

PDSS Definition, Procurement Practices, Personnel

Questions 1, 2, 3	Mark Levin, Chair Frank Moss, Scribe Pete Beck Dave Daniel Jan Grimes John LaVecchia Larry Lindley Jim Steenwerth
-------------------	--

Taxonomy of DOD Software

Questions 2, 4	Dan Kvenvold, Chair Steve Hudak, Scribe John Benson Ihor Hapij Karl Ipson Lou Naglak
----------------	---

PDSS Attributes

Question 5	Wes Babcock, Chair Ed Kutchma, Scribe Don Crocker Ray Day Roy Oldhan Bill Paine Dick Rubrecht
------------	---

These subpanels researched the issues and prepared the discussion, conclusions, and recommendations that follow. Summary interim reports of panel results were presented to the entire workshop using the briefing materials contained in Appendix D. Reference material used by the panel is contained in Appendix B.

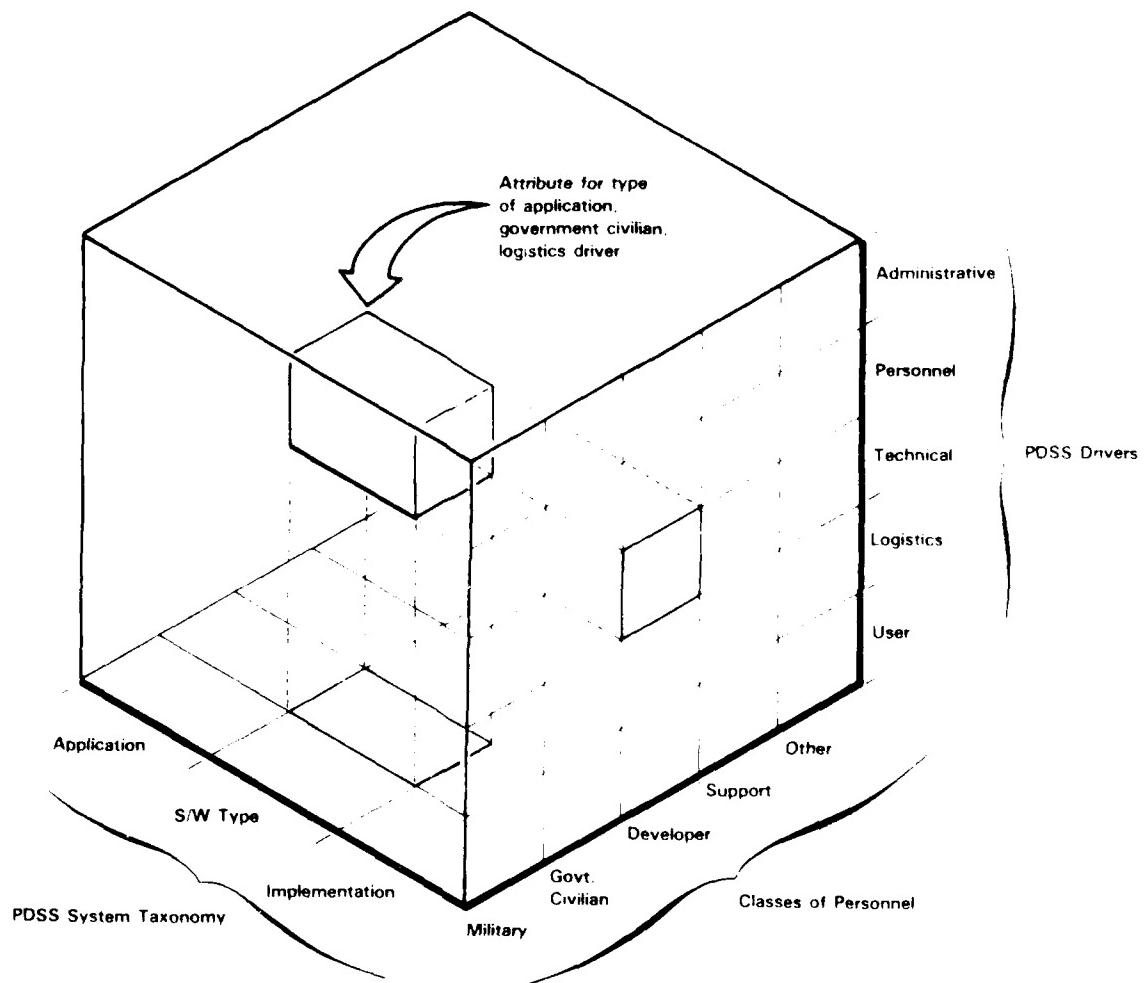


Figure 4.1-1. Interrelationship of Government/Industry Parameters

4.1.4 Discussion

4.1.4.1 PDSS Background

The entire system life cycle must be accounted for throughout all phases of the developmental process of a weapon system. Life cycle information must be prepared just after the conceptual stage commences. Usually the office which prepares such system information is that of the program/project manager. In a generic sense each service approaches this requirement in a similar fashion. The "plan" of action to accomplish the mission of developing/producing/fielding/ maintaining a system includes, if appropriate, a segment/plan on how the computer resources of the system will be acquired and managed. Each service has a regulation that purports to describe this plan. A new DID is being developed to prepare a common plan for all services in conjunction with MIL-STD-SDS. This plan is the Computer Resources Life Cycle Management Plan (CRLCMP). (The CRLCMP is the new MIL-STD-SDS term, and currently has counterparts in the Army's CRMP, the Navy's SLCMP, and the Air Force's CRISP.) The CRLCMP specifies the elements for both software development and PDSS of the system. It reflects all schedules, resource allocations, organizational interactions, and activity responsibilities associated with the project's life cycle.

What is apparent from a review of service management procedures is that there exist various regulatory mechanisms to accomplish the creation of a computer resource document. However, all services have problems in doing this work well:

- o Generally the document simply does not cover all the information it should.
- o The document either is not required early enough in a system's life cycle so as to impact on the resourcing of the system or is generated early and never updated.
- o The regulations which require computer resource documentation do not provide for sufficient discipline in the management process to ensure that the document is submitted as required.
- o The potential document users do not understand that such an early (in the system life cycle) management document must be a true living document, and hence do not plan for its being updated.

To properly plan a workforce mix that is attainable and achieves all goals for a given support system, the computer resource document must be on hand and must be used. Such a document, even as it evolves from one life cycle phase to another, must exist and must have widespread distribution among impacted activities/agencies. Coordinated and approved modifications to that document must, therefore, receive equal visibility and distribution.

The current DIDs that provide for PDSS are associated with CRLCMP, CRMP, SLMPC, or CRISP. There seems to be no modification of these DIDs that is required to organize and manage the PDSS. However, the panel unanimously concluded that these documents, if generated or developed, are thereafter either ignored or never updated as a program progresses. The result is that the PDSS tends never to be properly considered or planned at the time of system deployment.

Conclusion 1: The PDSS enabling documents (CRLCMP, CRMP, SLCMP, CRISP) should be properly filled out, with adequate consideration of the PDSS needs, and maintained throughout system acquisition.

What is the nature of PDSS? PDSS is a set of interrelated activities, products, and plans which differ from other life cycle activities, products, and plans because they focus on ensuring that software continues to satisfy its initial requirements after fielding. Proper support is essential to accomplish continuing satisfactory functional performance, continuing system reliability, and continuing supportability. This means that the error detection and correction process must be handled properly (not patches, but configuration-managed changes with compile and run). It also means that modification, for whatever reason (mission upgrade or enhancements), must not diminish the system integrity.

The word "deployment" in the term PDSS is of special concern because deployment marks the time at which software management responsibility is supposed to shift in most services. This time of responsibility shift is a function of system delivery, development completion, and other considerations. PDSS is not an activity that begins at a single point in time; but rather it is a collection of activities, each of which may begin at a point in time or which may be phased in over a period of time. While pre- and post-deployment technical activities are essentially the same, the management of these activities is different. Table 4.1-1 illustrates the gradual nature of the transition from pre- to post-deployment.

4.1.4.2 Definition of PDSS

As the government identifies the need for and procures an ever-increasing number of mission-critical computer systems, post-deployment software support must have a high priority in original planning. This is crucial if weapon systems which perform as intended are to be deployed. The panel concluded that a precise definition of PDSS was necessary to form the basis of a consistent policy on government/industry workforce mix, and it developed the following:

Conclusion 2: Post-deployment software support is the sum of all activities required to ensure that, during the production/deployment phase of a mission-critical computer system's life, the implemented and fielded software/system continues to support its original operational mission and subsequent mission modifications and product improvement efforts.

Table 4.1-1. PDSS Transition Points

<u>Factor</u>	<u>Pre-Deployment</u>	<u>Milestone</u>	<u>Post-Deployment</u>
Customer	Variable	Deployment	User
CM	Developer	Product baseline	Government
Money	R&D	Prod. decision or deployment	R&D/prod. R&D/prod./O&M
Proj mgt	Acquisition agency	Transition	Support
T&E Agency	DT&E	Variable	OT&E
Training	Developer	Variable	Government/ support

The term "software maintenance" was considered to be inadequate to convey the true nature of software support. Maintenance consists primarily of the activities and methods of restoring something that is broken to its original "unbroken" form. It is a term, derived from hardware, that conveys an erroneous picture of the true nature and complexities of software support. Software support is directed both at software redesign to correct software errors and at software design to enhance current features or to add totally new functions. "Software maintenance" simply does not convey either of these functions properly and therefore should not be applied to PDSS.

4.1.4.3 Description of PDSS Funding Difficulties

A problem of how PDSS is funded occurs commonly across the services. PDSS funding is almost always fragmented, making it difficult to manage properly. For example, Air Force system acquisition and PDSS are budgeted and funded through separate channels and processes (AFSC and AFLC). Even after program transfer, hardware and software are budgeted, funded, and prioritized by separate processes (BP 12, modification program and EEIC 583, engineering). Multiple budgeting and funding procedures exist for the same item, as opposed to separate budget and fund codes for separate but related items. Multiple procedures and fund codes can be used to acquire hardware, such as BP 8400, PRAM program, Project LIFT, capitalization of industrially funded operations, the Air Force Equipment Management System, and certain hardware and software supply fund codes. This creates confusion as to the proper acquisition process, clouds actual cost tracking, and requires careful coordination of one-year software money with three-year hardware money for the system modification. In some instances the PDSS responsibility is subdivided to both depot and field-level activities. This presents funding difficulties when planning to change the support base from organic to contract support. The problem is one of being unable to define the dividing line between depot and field responsibilities, and consequently the total funding and subsequent contracting responsibility default to the depot.

The Navy has similar problems in that a large portion of development and functional enhancement to a weapon system is done using Operational Maintenance, Navy (OMN) funds and Advance Procurement, Navy (APN) funds. If some funds are marked for multiple years and others must be obligated or outlaid within one year, contracting for PDSS tasks must be partitioned to accommodate this funding cycle. Task coordination and schedule interfaces become difficult, and delay or cost growth results. The proper allocation of dollars to functional tasks would improve the contracting posture and schedules of the PDSS function.

Conclusion 3: Streamlined policies and procedures are needed for budgeting and funding the development, acquisition, and support of mission-critical computer resources are needed. These should provide common budgeting and funding procedures among the services for presentation to the President and Congress, identification of appropriations, budget programs, program elements and specific fund codes to weapon systems, a single prioritization process, and simplification of procedures.

4.1.4.4 Workforce Utilization Assessment

The workforce elements used to perform PDSS are:

- o Government military personnel
- o Government civilian personnel
- o Original industrial software developer
- o Support services contractors that provide specific categories of personnel to perform engineering and technical services within the scope of a statement of work or under a level-of-effort or delivery order/tasking type of contract
- o Independent support contractor (independent of the original developer) that may be contracted with to provide total support of the software after deployment

The various permutations and combinations of industry/government workforce mixes can be summarized in three "most likely" PDSS organizations. There are, of course, many exceptions, including other non-US PDSS. The three that appear most viable are organic support, developer support, and independent support contractor (ISC) support. For organic support, PDSS is assigned to an organic activity within one of the military departments. The organic support activity reports to a system project manager and employs an optimum mixture of military, civil service, developer contract support, and/or support services contractors to accomplish the PDSS mission under the direction of the organic support activity. For developer (only) support, PDSS is contracted to the original developer for total PDSS support. The developer is contracted with by the system project manager, who directs the developer's PDSS efforts. In the final alternative, PDSS is contracted to an independent support contractor for total PDSS support. The ISC is contracted with by the system project manager, who directs the ISC's PDSS efforts. In all cases, regardless of which strategy is selected, the government must retain technical and managerial control of the PDSS.

Each of the classes of personnel was analyzed by the panel to determine how the mix impacts personnel attributes. The following attributes were used:

- | | | |
|-----------------------|-----------------|------------------|
| A. Cost | E. Training | I. Control |
| B. Stability | F. Experience | J. Security |
| C. Flexibility | G. Availability | K. Deployability |
| D. Relevant Knowledge | H. Continuity | |

To ultimately select the optimum PDSS workforce, many considerations should be addressed, each of which may favor a particular workforce option. An analysis was conducted for each of the above attributes. The results are given in Appendix E.

Conclusion 4: All system programs should analyze the specific system/mission requirements in the light of taxonomic and workforce option considerations. The analysis should identify the minimum set of management and technical functions which must be reserved to the government to assure PDSS control, to include:

- Project Management
- Financial Management
- Contract Management
- Technical Direction
- Acceptance/Rejection of Product
- Configuration Management
- Design Control Review and Approval

The output should be documented in the appropriate software life cycle management plan (e.g., CRLCMP) and reviewed and updated throughout the system life cycle.

4.1.4.5 Taxonomy of Mission-Critical Computer Resources

A strawman description of mission-critical software taxonomy was developed by the panel. It follows the six categories of software applications included under mission-critical system as prescribed in DoD Directive 5000.29 and the Warner Amendment.

The taxonomy hierarchy, structured into three levels to provide visibility to specific design applications that may require a particular support approach, is described in Appendix F. The Level 1 (general) category (i.e., weapon systems, intelligence systems, crypto and national security, command and control, direct support, and logistics) is further expanded to define the generic kinds of systems and software developed by the services within these categories. The Level 2 indenture describes the basic categorization of software according to its purpose (i.e., application, diagnostic, control, system-specific support, and vendor supplied). Level 3 identifies the implementation process by either software or firmware.

The initial planning phase for PDSS must consider the application categories in conjunction with the "attributes" to arrive at an efficient and cost-effective military, government-civilian, industry workforce mix. In general, pure taxonomy does not imply a specific workforce mix, with exceptions at Level 1 and with the general exception that any system that could require changes under combat conditions (e.g., submarines) may require military personnel. At Levels 2 and 3, there appears to be no generic reasons why a specific workforce mix is required, other than the obvious one of vendor software in Level 2.

The following system design trends will impact future mission/system techniques and methods. Their impact should be considered and planned in the design of PDSS concepts, facilities, and techniques.

- o Integrated and interoperable systems used for large data acquisition and fused information systems will require a more centralized PDSS for the integrated system, rather than support of individual subsystem software.
- o Standardized languages, software design tools, environments, and test tools will allow a generic PDSS to be used for several similar systems, resulting in utilization of common facilities and resources.
- o Commonality of software modules and software subsystems, resulting in reusable/validated software, will require an intersection and interface of the common module PDSS and each system PDSS.
- o Introduction of personal computers for use in the field, with users constructing their own software, will create an uncontrolled support problem. Configuration control must be addressed by some manner at system PDSS. Software should be properly documented and reviewed relative to its usage and limitations.
- o Artificial intelligence (AI)/expert systems will impact system design and result in crew function replacement by more integrated and interoperable software. This will in turn require more integrated and realistic approaches to software support, particularly the associated data interfaces and configurations. AI technology will also impact the automation of PDSS functions and facilities and should be planned for in a structured, automated life cycle environment.

4.1.4.6 Software Support Activity Drivers

The attributes that drive the selection of PDSS personnel were categorized into the following groups:

- o User Oriented
- o Logistics Oriented
- o Technically Oriented
- o Personnel and Resources Oriented
- o Administrative/Politically Oriented

For each group, subattributes were identified and analyzed for their impact on the workforce mix. The analysis of these subattributes is contained in Appendix G. The subattributes themselves are summarized in Table 4-1-2. It was felt that the driver attributes are a complete list. In application, some of them overlap, but this was not considered a problem for present purposes.

Table 4.1-2. POSS Driver Attributes

<u>User</u>		<u>Logistics</u>
U-1.	Wartime Support	L-1. Number of Unique Users
U-2.	Geographic Location	L-2. Deployment
U-3.	Embedded Doctrine/Tactics	L-3. Length of Life Cycle
U-4.	Responsiveness	L-4. Ancillary Requirements
U-5.	Interoperability	
U-6.	Field Support	
U-7.	Training PDSS Personnel	
U-8.	Criticality of System	<u>Personnel and Resources</u>
U-9.	Implementation Media	P-1. Special Facilities
U-10.	Change Process	P-2. Personnel Turnover
		P-3. Availability of Qualified Personnel
<u>Technical</u>		<u>Administrative/Political</u>
T-1.	S/W Configuration Management	A-1. Funding
T-2.	S/W Quality Assurance	A-2. Directed Procurement
T-3.	S/W Test Bed	A-3. Competitive Procurement
T-4.	Programming Languages	A-4. Personnel Ceilings
T-5.	Proprietary Software	A-5. Traditional Roles & Missions
T-6.	Adequacy of Documentation	A-6. International Support
T-7.	System Complexity	A-7. Security
T-8.	Software Engineering	A-8. Acquisition Management
T-9.	Software Maturity	A-9. Continuity of Operations
T-10.	Technical Risk	A-10. Commonality of Applications
T-11.	Built-in Test	
T-12.	Adaptability	A-11. Use of IV&V

4.1.4.7 Conclusions

The combined panel met to consider the usefulness of the above-described three-part analysis. It was decided to try a trial case on the model by selecting very different examples of the system taxonomy and seeing how applicable the attributes of support personnel and PDSS drivers would be. Three systems were selected:

- o Automatic Self Protective Jammer (ASPJ): A battlefield environment, operational system that is now being developed as a joint USN/USAF program.
- o USA Missleminder (AN/TSQ-73): A command and control system that has been operational and undergoing PDSS for several years. It employs a special-purpose computer and assembly language. Battlefield support is required (the special-purpose computer and language result in only a very small population of knowledgeable software people to do PDSS).
- o B-52 Weapon System Trainer: A support system being developed by the USAF using general-purpose computers and a common HOL.

Each of these three systems was evaluated using a metric of "+" to indicate that a particular class of personnel is either mandatory or highly desirable to satisfy the PDSS driver in question, and a metric of "-" to indicate that a particular class of personnel is either prohibited or is very undesirable to satisfy the PDSS driver. Where neither strong metric is applicable, the rating was left blank, indicating that this particular driver could be satisfied by any class of personnel. Tables 4.1-3, 4.1-4, and 4.1-5 summarize the panel findings for the three test cases. Several conclusions may be obtained from the three tables. First, an overwhelming number of attributes have no strong metric ("+" or "-") for the three projects. With a few exceptions, this indicates there is no a priori attribute that drives the assignment of a workforce. Second, a few attributes tend always to have the same result. For example, those attributes reflecting control (e.g., T-1, configuration management) would appear always to require either military or government civilian participation, independent of system taxonomy. Third, the less mature and more complex the system, the more participation is required by the original developer. Fourth, industrial participation by other than the original developer is driven only by either the need for additional staff (where military/civil service cannot be supplied) or by the need for lower cost (either original developer or civil service). From these observations, the following conclusions are drawn.

Conclusion 5: A low-level military presence is required to provide continuity and user influence and to govern embedded doctrine.

Conclusion 6: Government civilian personnel are required to provide technical capability as necessary to maintain government control and to provide an enduring corporate memory.

Table 4.1-3. ASPJ Model Analysis

Key: + Mandatory or highly desirable
- Prohibited or very undesirable

Drivers	Class of Support			
	Military	Govt Civilian	Original Developer	Support Contractor
User Drivers				
U-1. Wartime Support	+			
U-2. Geographic Location	-			
U-3. Embedded Doctrine/Tactics	+			
U-4. Responsiveness		+		
U-5. Interoperability			+	
U-6. Field Support				
U-7. Training PDSS Personnel				
U-8. Criticality of System				
U-9. Implementation Media	+			
U-10. Change Process	+			
Logistics Drivers				
L-1. Unique User				
L-2. Deployment				
L-3. Life Cycle Length				
L-4. Ancillary Requirements				
Technical Drivers				
T-1. S/W Configuration Mgmt				-
T-2. S/W Quality Assurance				-
T-3. S/W Test Bed				-
T-4. Programming Languages				-
T-5. Proprietary Software				-
T-6. Documentation Adequacy				-
T-7. System Complexity				+
T-8. Software Engineering				+
T-9. Software Maturity				+
T-10. Technical Risk				+
T-11. Built-in Test				+
T-12. Adaptability	+	+	+	+
Personnel Drivers				
P-1. Special Facilities		+		
P-2. Personnel Turnover	-		+	
P-3. Qualified Personnel Available				
Administrative/Political				
A-1. Funding				+
A-2. Directed Procurement				
A-3. Competitive Procurement				
A-4. Personnel Ceilings				+
A-5. Traditional Roles/Missions	+			+
A-6. International Support		+		
A-7. Security	+	+		+
A-8. Acquisition Mgmt		+		+
A-9. Operations Continuity	+	+		+
A-10. Applications Commonality	+	+		-
A-11. Use of IV&V				

Table 4.1-4. B-52 Weapon System Trainer Model Analysis

Key: + Mandatory or highly desirable
- Prohibited or very undesirable

Drivers	Class of Support			
	Military	Govt Civilian	Original Developer	Support Contractor
USER Drivers				
U-1. Wartime Support U-2. Geographic Location U-3. Embedded Doctrine/Tactics U-4. Responsiveness U-5. Interoperability U-6. Field Support U-7. Training PDSS Personnel U-8. Criticality of System U-9. Implementation Media U-10. Change Process	+			
Logistics Drivers				
L-1. Unique User L-2. Deployment L-3. Life Cycle Length L-4. Ancillary Requirements				
Technical Drivers				
T-1. S/W Configuration Mgmt T-2. S/W Quality Assurance T-3. S/W Test Bed T-4. Programming Languages T-5. Proprietary Software T-6. Documentation Adequacy T-7. System Complexity T-8. Software Engineering T-9. Software Maturity T-10. Technical Risk T-11. Built-in Test T-12. Adaptability			-	-
Personnel Drivers				
P-1. Special Facilities P-2. Personnel Turnover P-3. Qualified Personnel Available	-	-		
Administrative/Political				
A-1. Funding A-2. Directed Procurement A-3. Competitive Procurement A-4. Personnel Ceilings A-5. Traditional Roles/Missions A-6. International Support A-7. Security A-8. Acquisition Mgmt A-9. Operations Continuity A-10. Applications Commonality A-11. Use of IV&V	+	+		+

Table 4.1-5. AN/TSQ-73 Model Analysis

Key: + Mandatory or highly desirable
- Prohibited or very undesirable

Drivers	Class of Support			
	Military	Govt Civilian	Original Developer	Support Contractor
User Drivers				
U-1. Wartime Support U-2. Geographic Location U-3. Embedded Doctrine/Tactics U-4. Responsiveness U-5. Interoperability U-6. Field Support U-7. Training PDSS Personnel U-8. Criticality of System U-9. Implementation Media U-10. Change Process	+ +			
Logistics Drivers				
L-1. Unique User L-2. Deployment L-3. Life Cycle Length L-4. Ancillary Requirements				
Technical Drivers				
T-1. S/W Configuration Mgmt T-2. S/W Quality Assurance T-3. S/W Test Bed T-4. Programming Languages T-5. Proprietary Software T-6. Documentation Adequacy T-7. System Complexity T-8. Software Engineering T-9. Software Maturity T-10. Technical Risk T-11. Built-in Test T-12. Adaptability	-	-	+ + +	- - +
Personnel Drivers				
P-1. Special Facilities P-2. Personnel Turnover P-3. Qualified Personnel Available	-	+	+	+
Administrative/Political				
A-1. Funding A-2. Directed Procurement A-3. Competitive Procurement A-4. Personnel Ceilings A-5. Traditional Roles/Missions A-6. International Support A-7. Security A-8. Acquisition Mgmt A-9. Operations Continuity A-10. Applications Commonality A-11. Use of IV&V	-		+	- +

Conclusion 7: The original developer's participation is always required at fairly high levels on complex and immature software, and then dwindle as the software matures.

Conclusion 8: Support contractors provide services to obtain additional technical services not available through the government and to lower the cost of PDSS.

It is noted that the panel's determination of how government/industry personnel should be allocated does not markedly differ from the way the allocation is now generally made by the services. Certain minimum requirements exist that generally require on the order of 20% of the PDSS staff to be government and a number of staff to be supplied by the original developer (this number decreases as the software matures). The majority of the PDSS staff (approximately 80%) are then assigned from either civil service or industry based upon the particular needs/availability/funding or political outlook of the manager.

4.1.5 Recommendations

Recommendation 1: The JLC should enable procedures that ensure that the PDSS provisions in the new CRLCMP or its existing counterparts (US Army-CRMP, US Navy-SLCMP, and US Air Force-CRISP) are complied with at the outset of all software acquisitions, and ensure that the PDSS provisions are upgraded throughout program acquisition. This plan should be included at all service and DOD program reviews, including system acquisition review councils (e.g., DSARCs). (Refer to Conclusion 1.)

Recommendation 2: The JLC should streamline policies and procedures for budgeting and funding the development, acquisition, and support of mission-critical computer resources. (Refer to Conclusion 3.)

APPENDIX A
PANEL A PARTICIPANTS

PANEL A - INDUSTRY/GOVERNMENT WORKFORCE MIX

<u>CO-CHAIRMAN:</u>	Sisti, LTC F. (Frank) USA HQ DA (DAMO-C4L) Pentagon Washington, DC 20380	(202) 697-4539 A/V 227-4539
<u>CO-CHAIRMAN:</u>	Hartwick, Mr. R. D. (Dean) Logicon, Inc. 255 West 5th St. San Pedro, CA 90731	(213) 831-0611
<u>MEMBERS:</u>	Ipson, Mr. K. (Karl) USA CECOM DRSEL-SDSC-OD, Bldg. 1210 Ft. Monmouth, NJ 07703	(201) 532-5830 A/V 992-5830
	Beck, Mr. P. (Pete) AMCCOM/DRSMC-TSB(D) Dover, NJ 07801	(201) 724-4472 A/V 880-4472
	Hapij, Dr. I. (Ihor) USA ERADCOM (DELSD-SSC) Ft. Monmouth, NJ 07703	(201) 544-4741 A/V 995-4741
	Kutchma, Dr. E. (Ed) Naval Weapons Center (Code 05) China Lake, CA 93555	(619) 939-5230 A/V 437-5230
	Lindley, Dr. L. M. (Larry) Naval Avionics Center (Code D/072.2) Indianapolis, IN 46218	(317) 353-3979 A/V 724-3979
	Naglak, Mr. L. (Lou) Naval Air Development Center (50) Warminster, PA 18974	(215) 441-2314 A/V 441-2314
	Steenwerth, Mr. J. K. (Jim) Marine Corps Tactical Systems Support Activity Camp Pendleton, CA 92055	(619) 725-2607/2585 A/V 993-2607/2585
	LaVecchia, Mr. J. (John) WP-ALC/MMRC Robins AFB, GA 31098	(912) 926-4611
	Babcock, Mr. W. (Wes) SM-ALC/MMC McClellan AFB, CA 95652	(916) 643-4056

Hudak, Mr. S. (Steve) ASD/YWL Wright-Patterson AFB, OH 45433	(513) 255-7491 A/V 785-7491
Levin, Dr. M. (Mark) AD/ENE Eglin AFB, FL 32542	(904) 882-8505 A/V 872-8505
Crocker, Mr. D. (Don) Teledyne Brown Engineering Cummings Research Park Huntsville, AL 35807	(205) 532-1416
Day, Mr. R. (Ray) Intercon Systems 9400 Viscount, Suite 115 El Paso, TX 79925	
Moss, Mr. F. (Frank) Advanced Technology Inc. 1000 Paseo Camarillo, Suite 215 Camarillo, CA 93010	(805) 987-8831
Benson, Mr. J. L. (John) Chief of Airborne Software Bell Helicopter, TEXTRON Inc. P.O. Box 482, Dept. 81, M/S-6 Ft. Worth, TX 76101	(817) 280-3856
Oldhan, Mr. R. (Roy) TRW 708 Elberta Rd. Robins, GA 31093	(912) 929-1876
Grimes, Mr. D. (Dan) ADR Services Inc. 800 Follin Lane Vienna, VA 22180	(703) 281-2000
Rubrecht, Mr. R. (Dick) Flight Simulator Division Singer Company Binghampton, NY 13902	(607) 772-3917
Paine, Mr. B. (Bill) Jet Propulsion Laboratories Pasadena, CA 91109	(213) 354-4284
Daniel, Mr. D. (Dave) Naval Training Equipment Center (N-74) Orlando, FL 32813	(305) 646-4491 A/V 791-4491

APPENDIX B
BIBLIOGRAPHY

Bibliography Items:

1. AFLC Regulation 66-75, Subject: Depot Maintenance Source of Repair Decision Tree Analyses, 25 May 1979.
2. AFLC Regulation 400-26, Subject: Organic Versus Contract Decisions for Commercial or Industrial Activities, 19 December 1977
3. Army Regulation 18-1, Subject: Army Automation Management, 15 August 1980.
4. Army Regulation 70-1, Subject: Army Research and Development, 1 May 1973.
5. Army Regulation 70-17, Subject: System/Program/Project/Product Management, 11 November 1976.
6. Army Regulation 380-380, Subject: Automated Systems Security, 14 October 1977.
7. Army Regulation 1000-1, Subject: Basic Policies for Systems Acquisition, 1 May 1981.
8. Army Pamphlet 11-25, Subject: Life Cycle System Management Model for Army Systems, 21 May 1975.
9. Briefing Report, Subject: DoD Digital Data Processing Study - A Ten-Year Forecast, undated.
10. DARCOM Regulation 11-16, Subject: Program/Project/Product Management, 3 April 1979.
11. DARCOM Regulation 70-16, Subject: Management of Computer Resources in Battlefield Automated Systems, 16 July 1979.
12. DoD Directive Number 5000.1, Subject: Major Systems Acquisitions, 29 March 1982.
13. DoD Directive 5000.39, Subject: Acquisition and Management of Integrated Logistic Support for Systems and Equipment, 17 January 1980.
14. Military Standard Defense System Software Development, 30 July 1983, (Draft).
15. National Council of Technical Service Industries Report, Subject: Reliance on the Private Sector by the Federal Government for Data Processing Services, November 1974.

16. NAVAIR Instruction 5230.9, Subject: Policy and Procedures for the Establishment and Operation of Naval Air Systems Command Systems Software Support Activities, 14 June 1983.
17. OMB Circular Number A-109, Subject: Major System Acquisition, 5 April 1976.
18. Report of the DOD Joint Service Task Force on Software Problems, 30 July 1982.
19. USACSC Guidance for Software Development Contracting, undated.
20. USACOM Regulation 70-16, Subject: Policies and Procedures for Computer Resource Management (CRM) in Battlefield Automated Systems, August 1981.
21. USACSC Technical Bulletin, Subject: Resource Estimating Procedures for Software, 16 August 1982.

APPENDIX C
LETTER TO PANEL A PARTICIPANTS



DEPARTMENT OF THE ARMY
OFFICE OF THE DEPUTY CHIEF OF STAFF FOR OPERATIONS AND PLANS
WASHINGTON, DC 20310

REPLY TO
ATTENTION OF

DAMO-CMIL

29 AUG 1983

MEMORANDUM FOR MEMBERS OF THE INDUSTRY/GOVERNMENT WORKFORCE MIX
PANEL OF THE JOINT LOGISTICS COMMANDS (JLC) SOFTWARE SUPPORT CONFERENCE, ORLANDO I, 31 OCTOBER
1983 THROUGH 4 NOVEMBER 1983

SUBJECT: Panel Charter

We welcome you to the Industry/Government Workforce Mix Panel of Orlando I and thank you for having agreed to serve as a panel member. As you are aware, the workshop officially begins at 1:00PM on Monday, 31 October and continues through the week to a final session on Friday morning, 4 November. We will be the co-chairs of the Industry/Government Workforce Mix Panel.

Our panel has an ambitious topic for discussion. It is the goal of Orlando I that each panel should develop recommendations which would eventually be presented to the JLC for policy implementation. To attain that goal we will have to focus on our panel charter and prepare positions on the important questions prior to Orlando I.

We have taken the liberty of asking you to direct your attention to one of several major questions so that you can focus your thoughts and have a draft position on the question prepared to present to the panel at our first session on 31 October. This will mean some research will be in order, and we will need 23 copies of your draft position for the other panel members. Do not limit yourself only to the question to which you have been assigned. If you wish to prepare additional positions feel free to do so, but again bring 23 copies so we can all review them.

During the course of the week the panel will clarify and refine the positions which will then be molded into a coherent set of recommendations which will be presented during the 4 November wrap-up. We have attached a draft of our panel's charter which includes the basic issues, a number of questions and assigned position developers. We have tried to team up panel members on the questions so that the positions start with a broad base.

DAMO-C4L
SUBJECT: Proposed Charter

29 AUG 1983

Contact between subgroup members prior to Orlando I is highly recommended and contact with either or both of your co-chairs is also recommended. The more we use each other as sounding boards prior to Orlando I, the more we will be able to accomplish during the actual workshop.

Again, please accept our thanks for your participation and we are eager to hear from you and to work with you in Orlando.



FRANK SISTI
LTC, GS
HQDA, PENTAGON
DAMO-C4L
Washington, DC 20310
(202)697-4539
AV 227-4530



DEAN HARTWICK
LOGICON Inc.
255 W. 5th Street
San Pedro, CA 90731
(213) 831-0611

CHARTER OF PANEL ON
INDUSTRY/GOVERNMENT WORKFORCE MIX
IN
POST DEVELOPMENT SOFTWARE SUPPORT (PDSS)
FOR
EMBEDDED COMPUTER SOFTWARE

ISSUE: Once embedded computer products are developed and since the minimum governmental responsibilities include:

- o The performance of software configuration management
- o The retention of the respective software baseline
- o The review and approval of all change proposals for a system's software
- o Monitoring all applicable contracts,

What is the mix of government and industry participation in the PDSS of an embedded computer system?

SUB-ISSUES:

a. System Taxonomy - Each service approaches the support of automated systems based on specific parameters. An acceptable system taxonomy, based on agreed upon parameters, will greatly aid in establishing the Industry/Government Work Force Mix for system support.

b. Current PDSS Practices - PDSS is accomplished dependent on service and industry methodologies. A centralized listing of appropriate driving documentation and policies is required.

c. Roles in PDSS - The government and industry roles in PDSS for a system will define the appropriate Workforce Mix. The identification of an accepted role listing is required.

QUESTIONS: The following questions have been drafted for the purpose of generating panel discussion and clarification of the overall issue. They have been assigned to panel members (see list following questions) for research prior to the Orlando I Workshop. The questions are not listed in any order of priority.

- 1 What is the definition of PDSS, is it really a "Post Deployment" question?
- 2 What is the existing PDSS environment by service or industry.
- 3 What are the minimum role(s) of the government in PDSS beyond those given?
- 4 Is there a taxonomy of systems which lead to groups of systems that require separate developmental/sustainment, governmental/industry mixes?

- 5 What are the roles of the original equipment manufacturer and the initial support contractor in PDSS?
- 6 What current Government practices impact the balance of industry/government workforce mix; e.g., type of resource dollars, proprietary software and classified software.
- 7 What appropriate resource estimation tools/techniques exist for identifying required PDSS manpower/facilities?
- 8 Can industry "Fix-Forward", outside of the continental United States?
- 9 Where should PDSS be performed; e.g., in-theater, Government facility, contractor facility (response time is an important considerate).
- 10 How does the government ensure continued contract competition and new technology infusion?
- 11 Is a single workforce mix policy attainable?
- 12 Should the PDSS contractor be the OEM or an independent contractor not associated with the original ~~document~~
development?

PRIOR RESEARCH

Assignments

<u>QUESTION</u>	<u>PANEL MEMBER(S)</u>
1	Mark Levin & J. Steenwerth
2	Dr. Ed Kutchma & Dennis Turner
3	John LaVecchia
4	Wayne Bates, Steve Hudak, Lou Naglak, Thor Hapii
5	Roy Oldhan
6	Mickey Kincade, James Macdonell
7	Pete Eeck
8	Mel Dickover & Terri Payton
9	Ray Dav & Frank Moss
10	Bill Paine
11	Dan Grimes
12	Don Crocker

PANEL PRODUCTS:

- o A recommended system taxonomy for use in the categorization of embedded computer systems.
- o A compilation of the various PDSS efforts on-going in the services and applicable industry approaches.
- o A recommended policy on the Industry/Workforce Mix in PDSS of a system.

Panels will consist of membership assigned to the questions for initial sessions:

Panel A - Questions 1 & 4

Panel B - Questions 2, 3, 6, 7, 10

Panel C - Questions 5, 8, 9, 11, 12

ORLANDO I PANEL ON
INDUSTRY/GOVERNMENT WORKFORCE MIX
IN
POST DEVELOPMENT SOFTWARE SUPPORT (PDSS)
PROPOSED AGENDA

31 October 1983 (Monday)

- o (AM) Registration
- o (PM) Key note address
- o (Evening) Joint Mixer

1 November 1983 (Tuesday)

- o Joint chairs meet
- o (AM) Panel meeting
- o (Lunch) Guest speaker
- o (PM) Definition of issue consensus
- o Assignment of issues to sub panels
- o 4:30-5:30PM Joint Panel meeting

2 November 1983 (Wednesday)

- o Joint chairs meet
- o (AM) Sub panel meetings
- o (Lunch) Guest Speaker
- o (PM) Sub panel meetings
- o 4:30-5:30PM Joint Panel meeting
- o (Evening) Joint Dinner

3 November 1983 (Thursday)

- o Joint chairs meet
- o (AM) Sub panel 1, 2, 3 presentations
- o (Lunch) Guest speaker
- o (PM) Collective sub panel meeting
- o Panel consensus meeting
- o 4:30-5:30PM Joint Panel meeting

4 November 1983 (Friday)

- o (AM) Joint meeting to present final panel recommendations

APPENDIX D
PANEL A BRIEFINGS

INDUSTRY/GOVERNMENT WORKFORCE MIX

SCOPE: CONVERGE ON A SINGLE POLICY FOR SUBMISSION TO JLC

- ISSUES:
- o WHERE SHOULD PDSS BE PERFORMED
 - o WHAT ARE POLICY DRIVERS
 - TECHNICAL
 - PROCEDURAL
 - SECURITY
 - COST
 - RESOURCE AVAILABILITY
 - LOGISTICS
 - o WHAT ARE SERVICE ORGANIZATIONAL AND POLICY CONSTRAINTS

INITIAL BRIEFING

PANEL A METHODOLOGY

REVIEW PANEL CHARTER

DEFINE ISSUES

ESTABLISH SYSTEM TAXONOMY

MODEL POLICY DRIVERS

- TECHNICAL,
- PROCEDURAL,
- SECURITY
- COST
- RESOURCE AVAILABILITY
- SERVICE POLICY
- LOGISTICS

DEVELOP JLC POLICY POSITION

DOCUMENT FINDINGS & RECOMMENDATION

PANEL A AGENDA

<u>MONDAY</u>		
1615 - 1730	REVIEW PANEL CHARTER DISCUSS EXPECTED RESULTS REVIEW PANEL AGENDA REVIEW PDSS DEFINITION & EXISTING PRACTICE	
<u>TUESDAY</u>		
0800 - 1200	ESTABLISH PANEL ISSUES ASSIGN ISSUES TO SUBPANELS	
1200 - 1330	LUNCHEON	
1330 - 1615	SUB-PANEL SESSIONS	
1630 - 1730	JOINT SESSION	
<u>WEDNESDAY</u>		
0800 - 0900	SUBPANEL PRESENTATIONS - SUBPANEL APPROACH	
0900 - 1200	SUBPANEL SESSIONS	
1200 - 1330	LUNCHEON	
1330 - 1515	SUBPANEL SESSIONS	
1515 - 1615	SUBPANEL PRESENTATIONS - PRELIMINARY RESULTS	
1630 - 1730	JOINT SESSION	

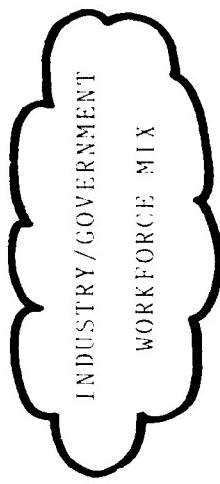
PANEL A AGENDA (CONTINUED)

THURSDAY

0800 - 1000 SUBPANEL SESSIONS
1000 - 1200 PRESENTATIONS - FINAL RESULTS
1200 - 1330 LUNCHEON
1330 - 1630 SUBPANEL - DOCUMENT RESULTS
1630 - 1730 JOINT SESSION - PANEL SUMMARIES

FRIDAY

0800 - 0945 JOINT SESSION - PANEL SUMMARIES
1000 - 1200 JOINT SESSION - WRAP-UP



SCCIE: CONVERGE ON A SINGLE POLICY FOR SUBMISSION TO JLC

- ISSUES:
 - o WHERE SHOULD PDSS BE PERFORMED
 - o WHAT ARE POLICY DRIVERS
 - TECHNICAL
 - PROCEDURAL
 - SECURITY
 - COST
 - RESOURCE AVAILABILITY
 - LOGISTICS
 - o WHAT ARE SERVICE ORGANIZATIONAL AND POLICY CONSTRAINTS

PANEL A
PROCEDURE

1. 20 PAPERS SUBMITTED ON 12 ISSUES
2. REVIEWED STATEMENT OF CHARTER AND PROPOSED
RESTATEMENT FOR COMPLETENESS
3. PRESENTATIONS OF CURRENT PDSS ENVIRONMENT FROM
FOUR SERVICES
4. REVIEW OF AGENDA AND SCHEDULE

PANEL "A" SUB-PANEL
ON
PDSS FUNDING ISSUES

1. IDENTIFYING IT AND RAMPING IT INTO POM SUBMISSIONS
2. PRIORITIZING IT HIGH ENOUGH THAT IT GETS IN
3. PROTECTING IT FROM INCURSIONS BY OTHER PROGRAM ACTIVITIES
4. (AF ONLY) COORDINATING SUPPORTING COMMAND REQUIREMENTS FOR R&D FUNDS

DRAFT

PANEL, "A" SUBPANEL

ON

PDSS DEFINITION

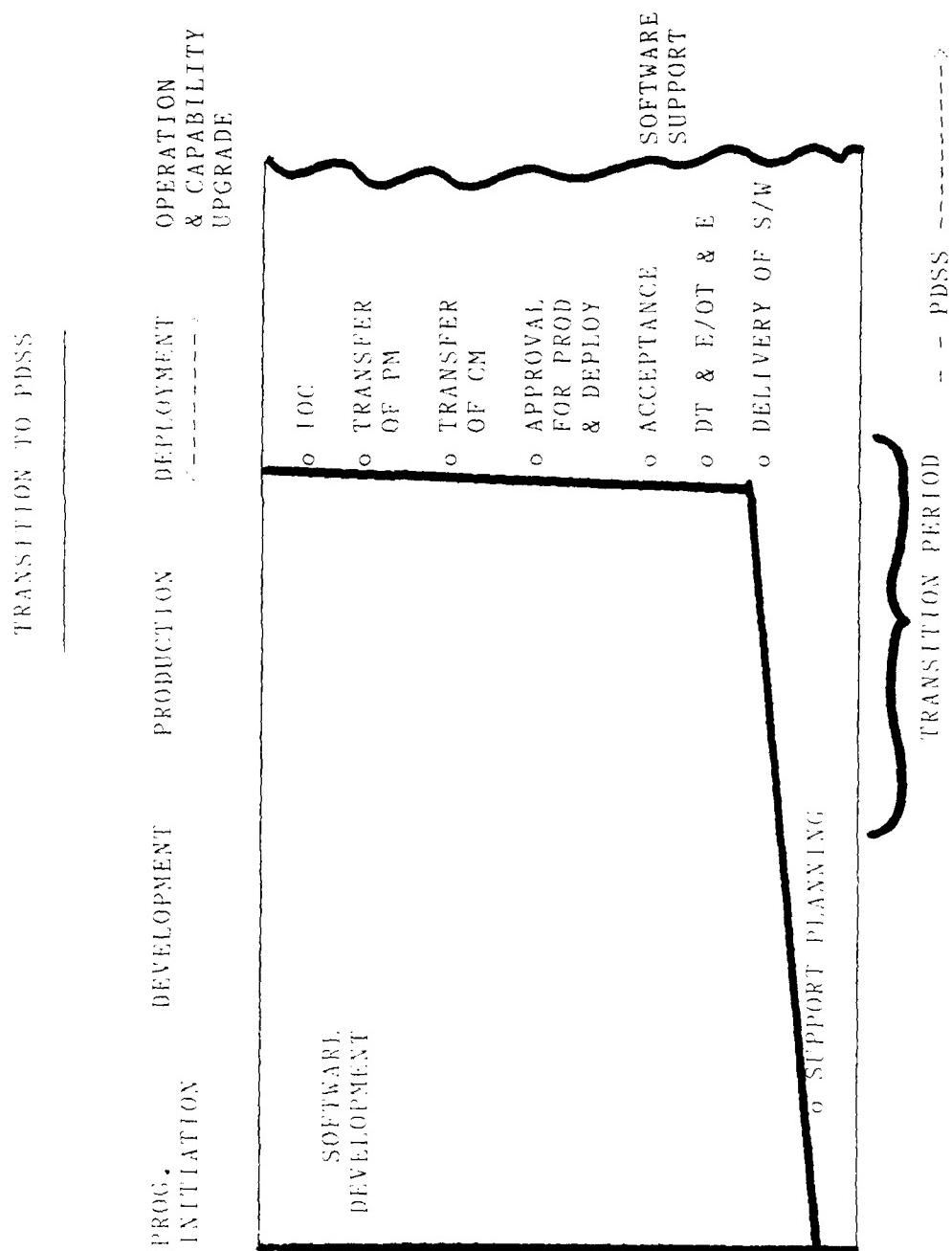
DRAFT

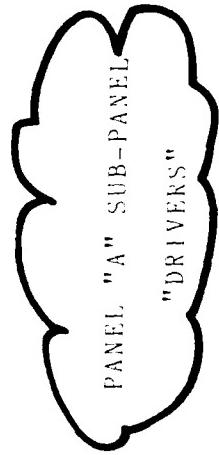
SOFTWARE SUPPORT IS THE SUM OF ALL THOSE ACTIVITIES WHICH TAKE PLACE DURING THE SYSTEM'S LIFT, FROM REQUIREMENTS DEFINITION THROUGH COMPLETION OF ITS LIFE CYCLE, FOR THE PURPOSE OF ENSURING THAT THE IMPLEMENTED AND FIELDED SOFTWARE CONTINUES TO FULLY SUPPORT THE OPERATIONAL MISSION FOR WHICH IT WAS ORIGINALLY CONCEIVED AND DESIGNED.

"ANON"

PANEL "A" SUB-PANELS
ON
PDSS ENVIRONMENT

TAXONOMY		"DRIVERS"			
		LOGISTICS	TECHNICAL	PER. RES.	POL. CON.
PERSONNEL MIX	USER				
MIL.				PRESENT FUTURE PREFERRED	
GOVT.					
GOVT/CIV					
OEM					
IND					
SUPPORT					





CONCLUSIONS:

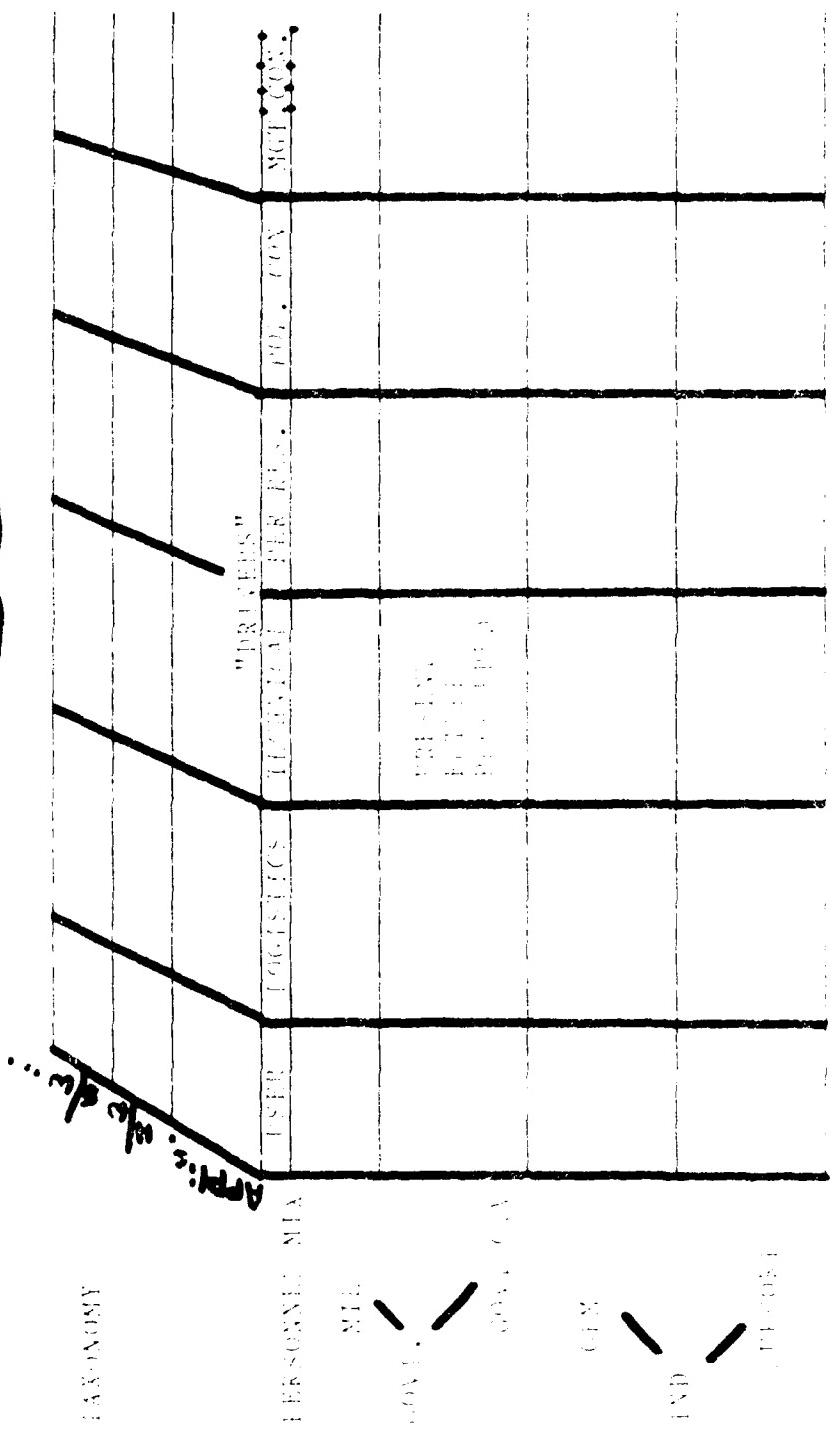
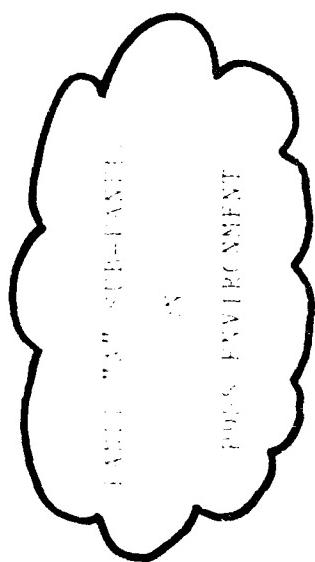
- o SOFTWARE PLANNING
 - USE CUBE
 - DONE EARLY
 - LIVING DOCUMENT
- o HEURISTICS ARE DUMB
- o CURRENT PDSS
 - FUNDING
 - PERS. AVAIL..
- o NO NEW PLANS/REVIEWS
- o SERVICES IN SAME BOAT

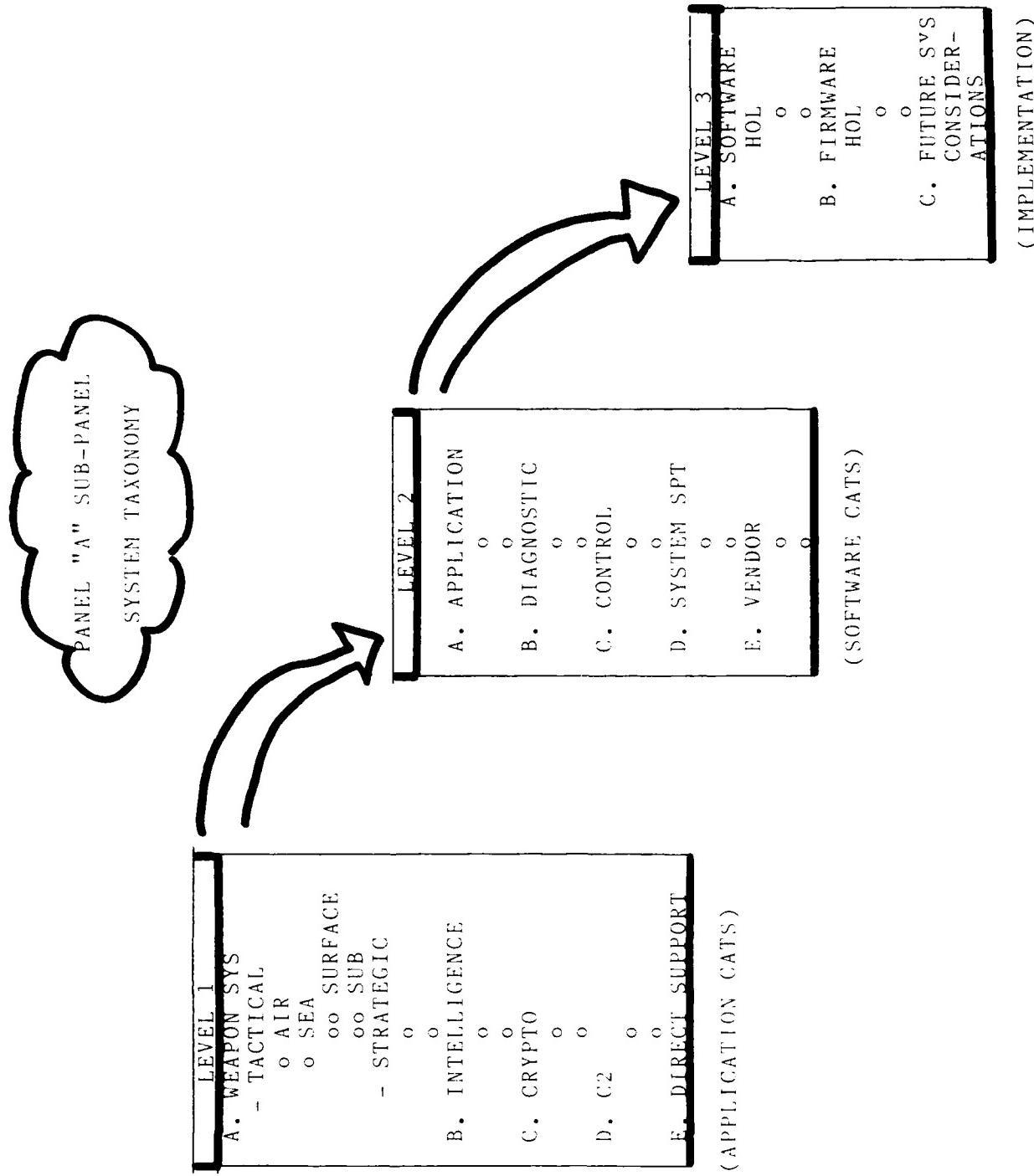
PANEL "A" SUB-PANEL.

"DRIVERS"

RECOMMENDATIONS:

- o ENCOURAGE EARLY, EFFECTIVE SOFTWARE PLANNING
- o STRONG LINKAGE
PLANS - P.M.
- o USE CUBE TO IMPROVE PLANNING

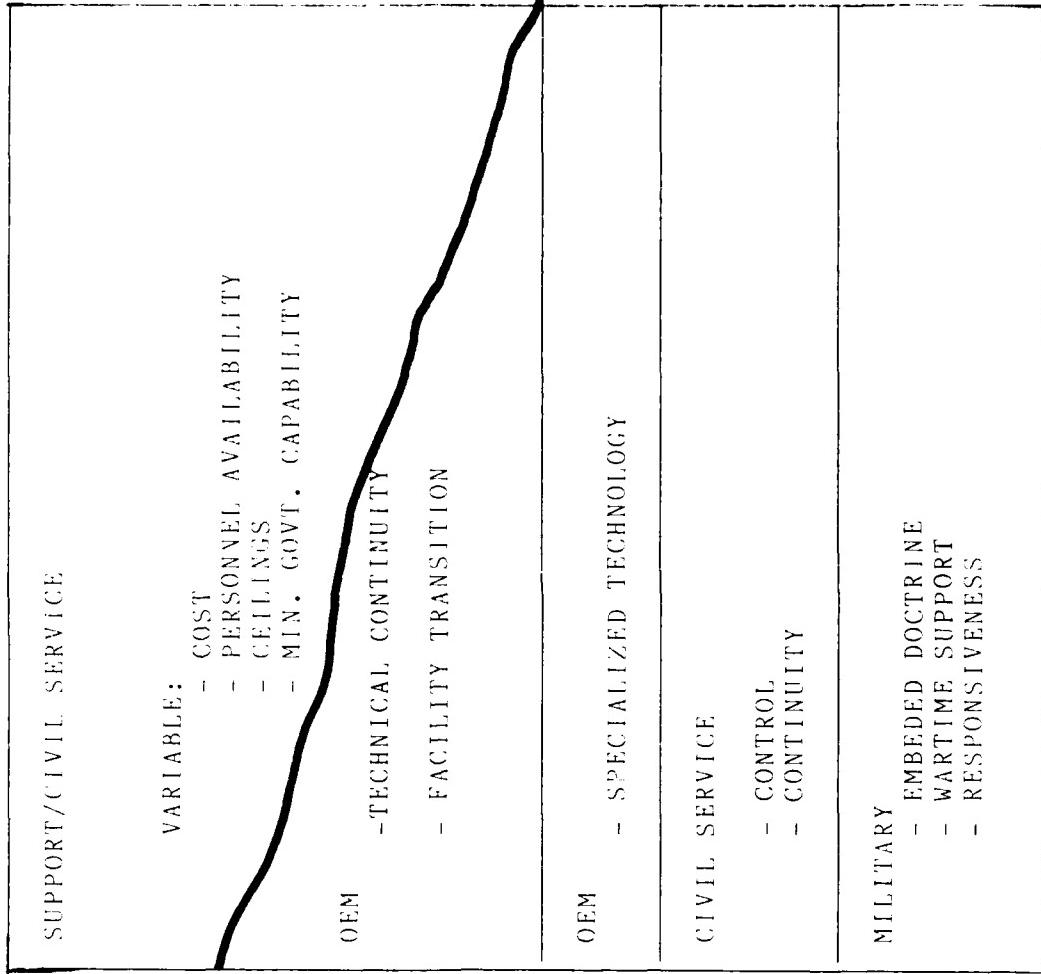






1. ESTABLISH COMMON PDSS DEFINITION
2. ESTABLISH COHERENT PDSS FUNDING POLICY
3. DEVELOP MINIMUM ESSENTIAL PERSONNEL GUIDELINES
4. DEVELOP "HOW-TO" AID FOR CRLCMP (CRMP, SLCMP, CRISP)
5. IMPLEMENT "LIVING DOCUMENT" FEATURES OF CRLCMP

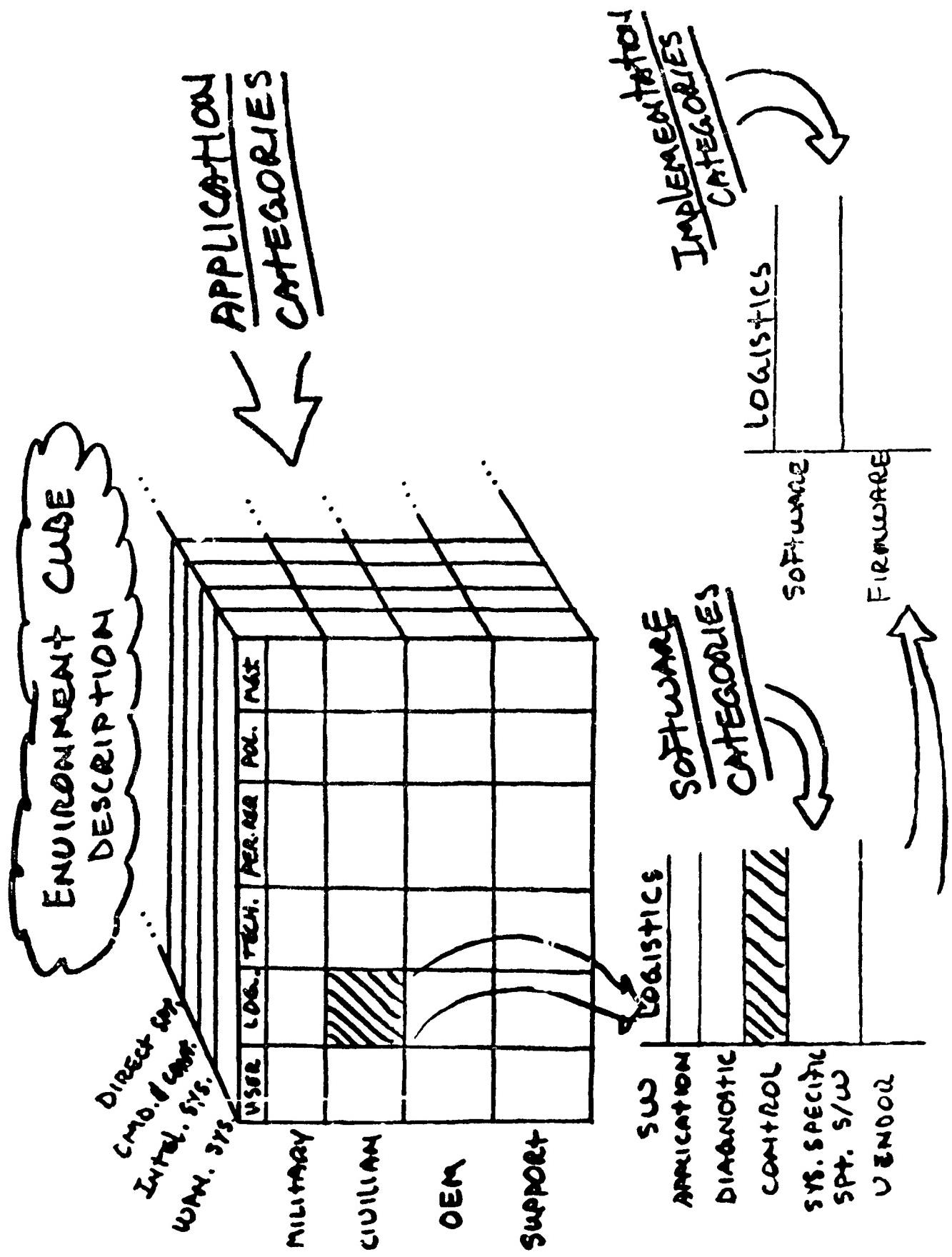
PERSONNEL
ALLOCATION

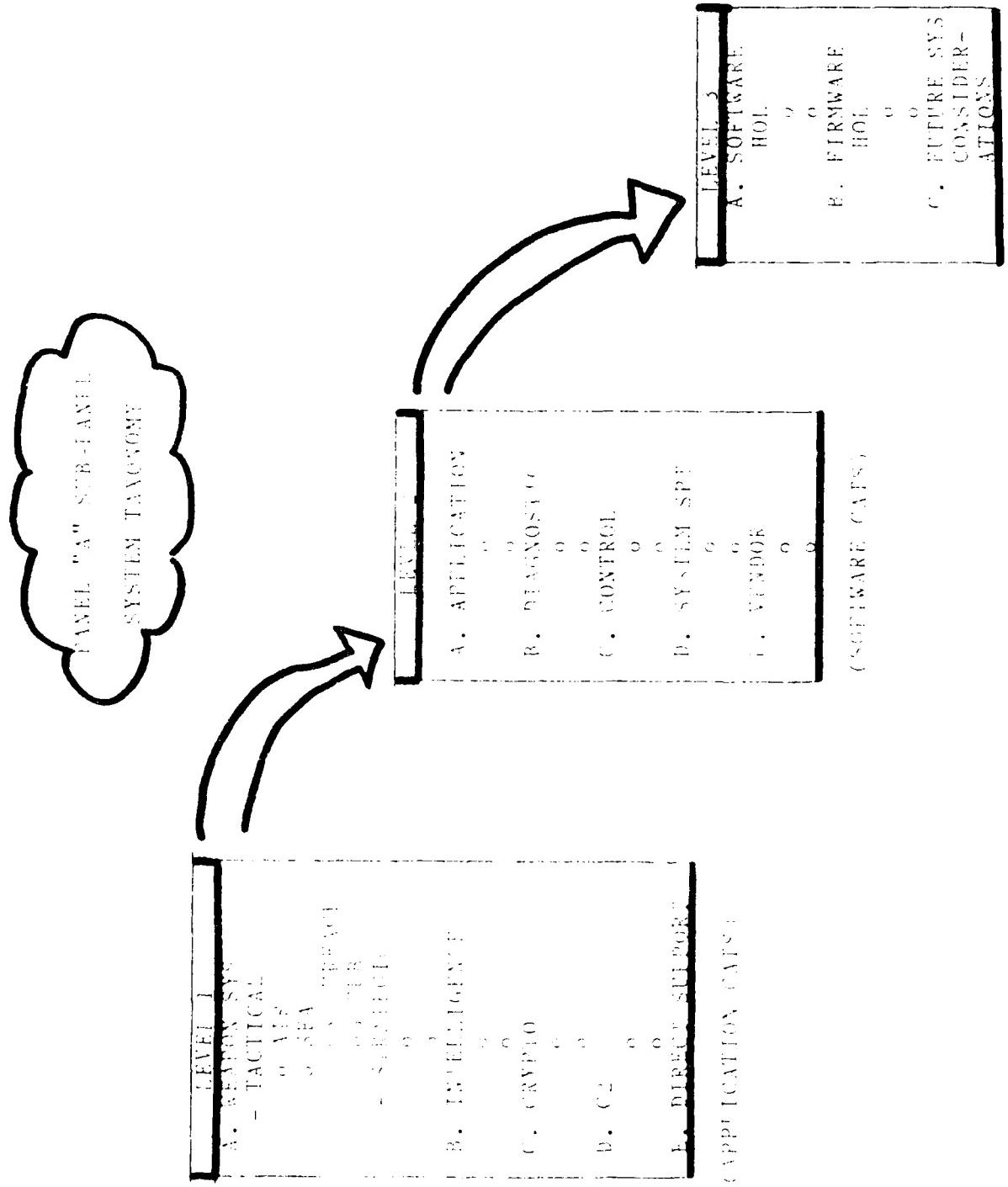


VARIABLE

CORE

TIME - - - - -





MIX CONSIDERATIONS:

COST (OF PAX)

STABILITY (OF PAX)

FLEXIBILITY (OF PAX)

RELEVANT KNOWLEDGE

TECHNICAL KNOWLEDGE

OPERATIONAL KNOWLEDGE

TRAINING CONSID.

EXPERIENCE CONSID.

MIX CONSIDERATIONS:

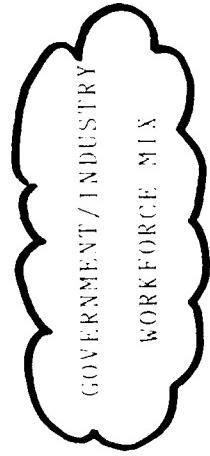
AVAILABILITY (OF PER TYPES)

CONTINUITY (OF PER OVER TIME)

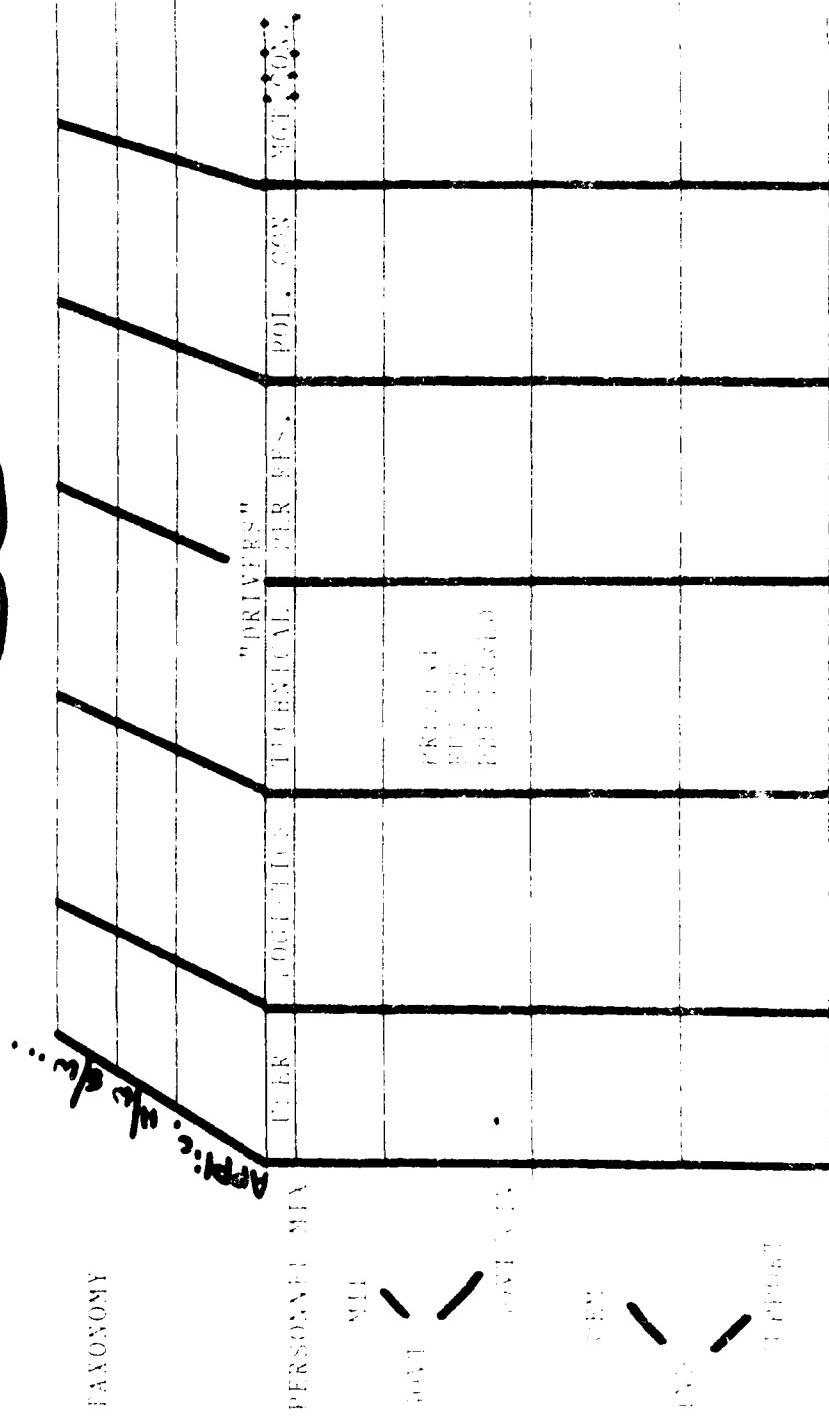
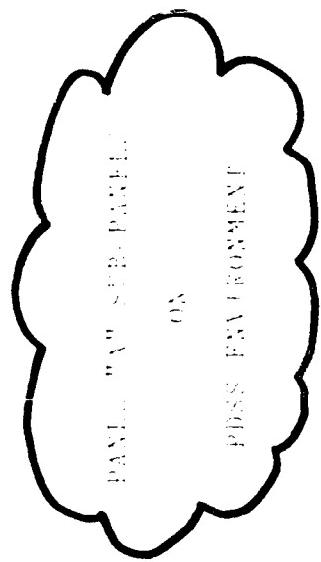
CONTROL (WHO SHOULD DO WHAT)

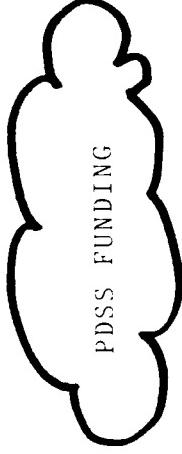
SECURITY (WHO IS MOST TRUSTWORTHY)

DEPLOYABILITY (WHO CAN GO)



- o MINIMUM GOVERNMENT RESPONSIBILITIES
- o MIX RECOMMENDATION
- o MIX CONSIDERATIONS



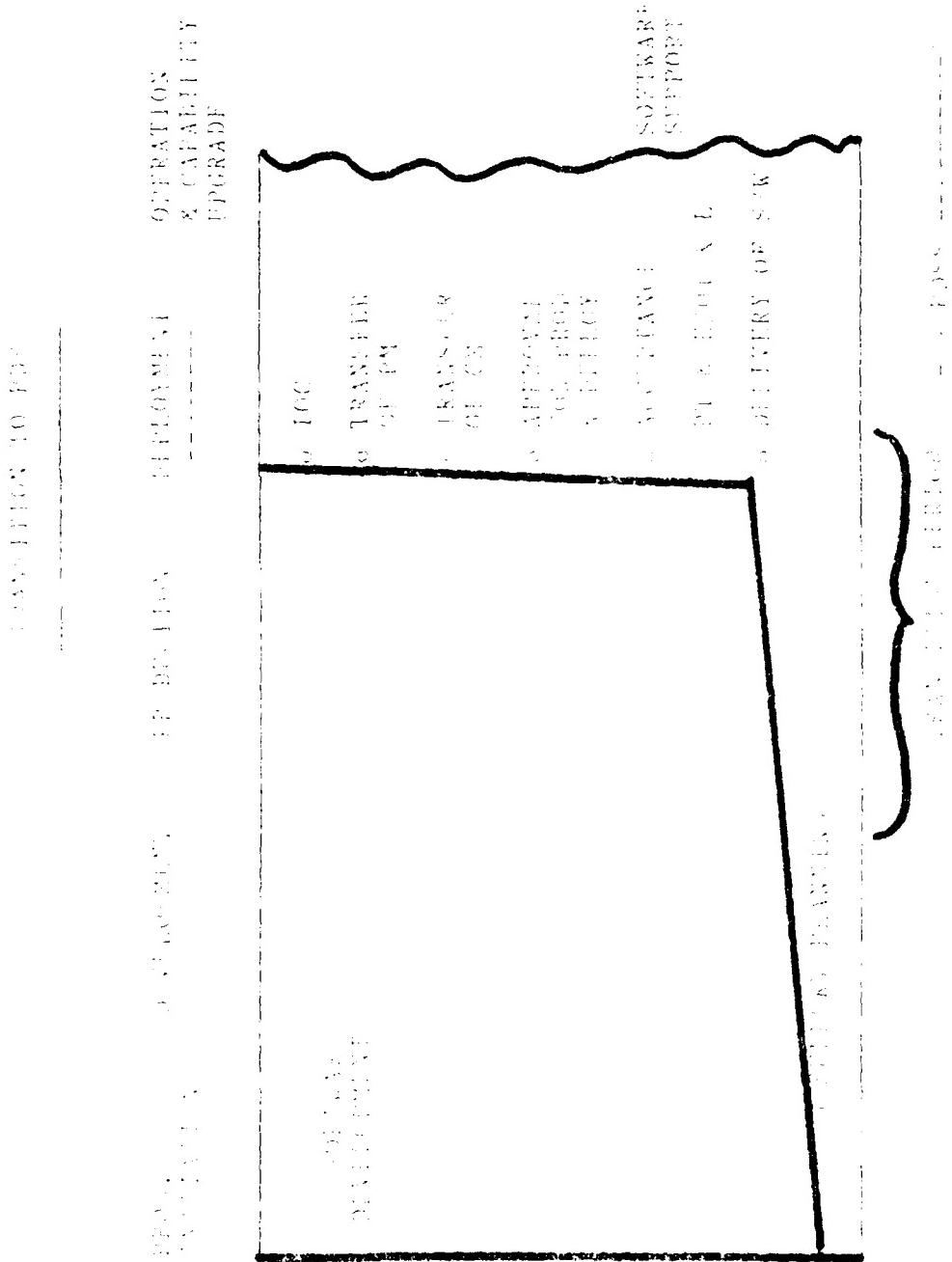


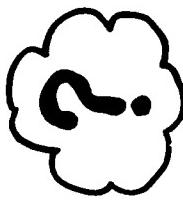
ISSUES:

- IDENTIFYING WHAT RESOURCE DEMANDS ARE TO BE MET
- IDENTIFYING WHAT ACTIVITY SHOULD PLAN / REQUEST FUNDING
- IDENTIFYING PROPER ACQUISITION CHANNEL

CONCLUSION:

LIMITED PARALLEL ACROSS SERVICES REQUIRE COHERENT APPROACH





DOES A COST DEPLOYMENT PROBLEM?

- WHAT IS PROBLEMS?
- WHAT IS ... ?
- MAINTENANCE PLANS

INDUSTRY/GOVERNMENT WORKFORCE MIX

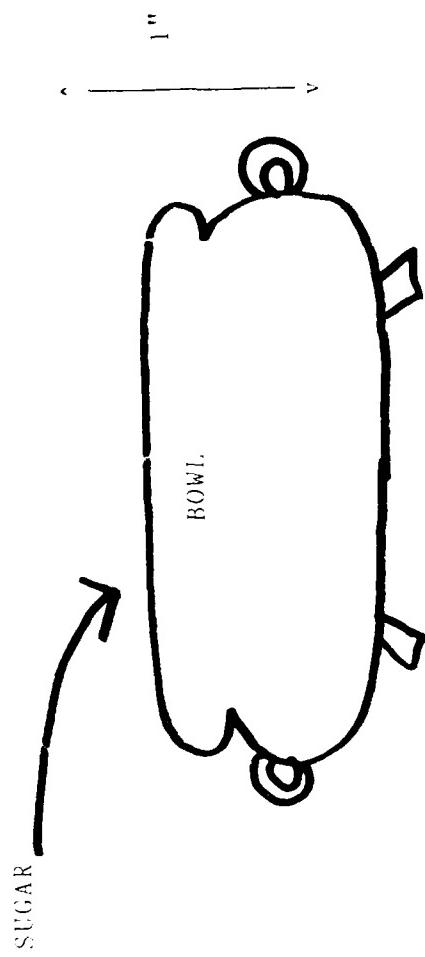
SCOPE:

RECOMMEND POLICY FOR
DETERMINING PERSONNEL
MIX FOR DOD SOFTWARE
SUPPORT AGENCIES

ISSUES:

1. WHAT IS PDSS?
2. WHAT ARE PDSS FUNDING PROBLEMS?
3. WHAT ARE QUALITATIVE DIFFERENCES IN
PERSONNEL MIXES?
4. WHAT EFFECT DOES SYSTEM TAXONOMY HAVE?
5. WHAT DRIVERS AFFECT THE MIX?

PANEL A
FINDING



APPENDIX E
WORKFORCE ATTRIBUTE ANALYSIS

WORKFORCE ATTRIBUTE ANALYSIS

A. Cost: Cost is usually the major consideration in organizing to perform PDSS and must be weighed carefully against each of the other PDSS factors. Costs vary across the major categories of the PDSS personnel sources, i.e., government military and civilians, the domestic software industry in the United States, a mix of both government and industry personnel. In addition, there are lesser, but significant and growing, sources outside the United States.

In determining the best source or mix of personnel, the annualized total life cycle costs of these personnel must be determined rigorously and projected across the expected PDSS years. The PDSS years may extend, depending on the system and mission, from eight to over thirty years!

- o Government: In determining the PDSS costs for government-military and government-civilian personnel, the life cycle costs of an engineer at the doctoral level vary significantly from a technician's. Using averages that cross the knowledge and skill levels for simplicity of computation, then spreading these figures across thousands of personnel and several decades can lead to costing errors in the billions of dollars!
- o Industry: In determining the costs of employing contract personnel, equally misleading cost projections can be developed by unnecessarily:
 - Using estimates when firm bid figures can be solicited from industry
 - Requesting bids from only a single contractor or a small set of contractors
 - Lumping all PDSS into one very large, long-term contract
- o Mix: It can be expected that a mix of government-military and civilian and one or several US contractors will frequently yield lesser PDSS cost figures than either a dedicated government center or a sole-source contract. To ensure satisfactory performance, the least-cost mix will need to be adjusted so that all other factors are satisfactory.

In summary, cost projections must identify full life cycle costs associated with the different professional and technical levels in the government. Costs to be incurred by PDSS activities using contract support should be derived from real bids submitted by firms to perform specific PDSS tasks or to provide personnel with specific expertise over the systems life. These same principles should be applied whenever PDSS can be supported by other (non-US) sources. With realistic cost figures, PDSS managers can plan and organize for least cost and equally effective support.

B. Stability:

- o Government: Military rotation problems and enlistment termination are problem areas. Civilians would appear to be the most stable, but current competition from industry and promotional opportunities lead to employee mobility. Thus stability is not in fact assured. Also government recruitment and retention are problems.
- o Industry: The original developer personnel are in general unknown, but as development is usually a company thrust, personnel initially supporting the system in an interim support period capacity may tend to return to the home base for a subsequent development program. Many independent software support contractors are beginning to surface, and their capabilities are becoming significant. This may provide a stable workforce element, especially for companies that are totally service oriented. An important consideration is the competitive posture of the government. If the government competes periodically for the software support, alternate ISCs may win, thus leading to support instability. However, personnel working for one ISC may elect to change companies in lieu of moving. In this case both the winning ISC and the individual win as the cost and the pain of relocation is avoided.
- o Mix: Government management control retention, corporate technical knowledge, technical direction, and facility and baseline ownership with varying degrees of support provided by industry (depending on immediate circumstances) appear to be best for ensuring "continuity," though stability can still not be quantified nor assured. There are essential roles that the government must retain in order to ensure continuing support capabilities at reasonable cost.

C. Flexibility: Flexibility incorporates the ability to increase or decrease the number of qualified people in response to increases or decreases in the workload.

- o Government: Flexibility is low, due to difficulties in hiring, firing, promoting, or demoting people. These difficulties are due to personnel and grade level ceilings, salary structures, additional personnel goals (such as protection against political pressures, EEO, refraining from competition with industry, etc.), and other similar factors.
- o Industry: Flexibility is higher, due to the factors noted above, and also due to greater ease in letting subcontracts and using contracted temporary help.

- o Mix: The greater flexibility inherent in industry is also available in a mixed environment, since the tasks which are most variable in workload (designing, making, and testing software charges) are those which are generally assigned to the contractor.
- D. Relevant Knowledge: Knowledge of the software system and the total mission-critical system is required for effective PDSS. Knowledge is required in technical areas, as well as an understanding of the software and the system performance, operational use, and support.
 - o Government: Technical knowledge level can be high with an investment in theoretical and "hands-on" training on the system and the software at the original developer's plant. Operational knowledge concerning use and support of mission-critical systems is resident within the military forces and, to a lesser extent, in the civil service workforce. The military must provide final guidance in this area.
 - o Industry: Technical knowledge of the original developer is high. Technical knowledge of independent support contractors or support service contractors in the specific system to be supported is highly variable. Developers possess a variable level of operational knowledge depending on the original developer's experience with similar operational systems. Support service contractors often have high levels of operational knowledge, since many are often heavily staffed by former military personnel who collectively possess a wealth of operational knowledge. Caution is required to prevent these contractors from usurping military decisions regarding operational use and support.
 - o Mix: Requisite technical knowledge of specific systems may best be provided by a judicious mix of Government, original developer, and support service contractors. Operational knowledge must be provided by military user community representatives, with the option of augmentation by qualified support service contractors.

E. Training:

- o Government: Training for military personnel, especially for enlisted personnel, is a time-consuming and costly process with a high attrition rate. GS personnel (engineers and computer scientist types) are usually hired with fundamental training completed. Further training with long-term payoffs is considered to be money well spent.
- o Industry: Industry personnel theoretically arrive on site trained for their specific functions. Further training occurs

on the job; and for professionals, personal development generally follows due to self-motivation.

- o Mix: A mix allows training of an essential cadre of government specialists to provide management and technical direction for continuity and to maintain a sound government competitive posture with a supply of talent for task performance from industry.

F. Experience: PDSS requires both varied experience factors in multiple disciplines and the sustaining of a critical mass of experience factors and professional and technical expertise related to the evolving system software.

- o Government: Experience in using the mission-critical system in military operations must be organic to the PDSS organization. Recent or current experience can only be provided by military personnel across the enlisted, company, and field grade levels and across the occupational specialities associated with the system as it evolves. Experience in operations and technical areas is brought to PDSS by the government civilian workforce. Experience and knowledge in working with service formal and informal command, support, and information channels are almost unique to government civilians. The technical and professional experience in many of the disciplines required to perform PDSS are also resident in the government workforce. However, PDSS requirements exceed the military and government resources which can be devoted to PDSS over the system life cycle.
- o Industry: By drawing on personnel who have served in the military, industry can provide relevant operational experience to most systems. However, industry contribution in operational experience is not as certain as that in the active military. Industry can provide the most experience in all technical and professional disciplines required to perform PDSS. Industry can move experience as required across systems, subsystems, and the services as required. Because contractors operate in a highly competitive environment, they have to maintain state-of-the-art knowledge of the widest range of hardware and software products and evolving concepts in, and coming into, the marketplace. This experience facilitates system evolution and technology insertion in PDSS activities.
- o Mix: To provide the multiple operational and technical experience factors required to cost-effectively perform PDSS over a system's life, a mix of personnel drawn from each of the workforce options will generally be required. The composition of the mix will have to be carefully derived for each PDSS to optimize:

- Reusability of software already inventoried
- Redesign
- Testing
- Software performance in operation

In deriving the mix, the PDSS manager must identify in detail a comprehensive list of operational and technical experience factors to perform PDSS over the system's life. Generally, to conserve government resources, technical experience should be drawn from industry. Operational experience factors will have to be drawn from government sources. Positions must be described using the specific required experience factors. These should describe in detail the mission, functions of the hardware, and all of the systems and subsystems requiring software and their specific architecture, versions, or releases.

The development of the requisite experience factors to perform a particular PDSS should be performed before costing or other PDSS decisions are finalized. Personnel to meet these requirements can be drawn from government or industry. It can be expected that a mix including military and government civilians will be required just to provide the necessary operational experience. Industry can provide the mass of the technical expertise required. Identification of the specific, detailed experience factors is the first and driving consideration in establishing an organic PDSS workforce.

G. Availability: Availability of workforce personnel having the requisite skills is a continuing problem. Critical software engineering and systems engineering skills shortages will continue throughout the 1980s. The ability to attract and retain a workforce with these requisite skills is highly dependent on the workforce option.

- o Government: The military and civil service communities have experienced severe difficulties in attracting and retaining "working-level" software engineers, computer scientists, and systems engineers. Upcoming changes in the civil service retirement structure will exacerbate this problem. Another constraint on some government activities is the availability of billets or ceilings, or grade-point average limitations which prevent optimum civil service staffing. The availability of properly qualified military personnel to fill open military billets is a continuing problem.
- o Industry: The original developers have the greatest success in attracting the critical technical skills, primarily because they can outbid all others. Support services contractors also can attract and retain these critical talents, but to a lesser extent than the developer since they must be price "competitive" with the civil service workforce.

- o Mix: The availability of sufficient requisite skills is best assured by providing a mix of military, civil service, and industry personnel.
- H. Continuity: Continuity implies an ability to remember past mistakes and the background to recall insufficiently documented reasons for the rejection of possible solutions. Stability enhances this process, but is not necessary for it. A lack of stability can be compensated for by careful, complete documentation, or by continued access to personnel no longer actively working on a project.
 - o Government: Continuity varies from organization to organization but tends to be relatively high due to fairly high stability.
 - o Industry: Continuity varies even more than for government, due not only to variations in stability, but also to variations in corporate practices. If a corporation is the original developer, significant additional continuity is obtained.
 - o Mix: In general, continuity will be dependent on the continuity in the government participation and the presence or absence of the original developer in the mix. It is also dependent on the stability of the mix itself, in terms of what companies are involved.

- I. Control: All actions necessary to manage, direct, monitor, or control PDSS activities are encompassed within this workforce consideration. The key attributes or characteristics of effective PDSS control are:

- o Project Management
- o Financial Management
- o Contract Management
- o Technical Direction
- o Acceptance/Rejection of Product (Acceptance Agent)
- o Configuration Control
- o Design Control (Design Review/Approval---Design Agent)

It is suggested that the characteristics listed above are a minimum set of "features" which are necessary to ensure effective PDSS control.

- o Government: The government must reserve to itself the control of all activities during PDSS.
- o Industry: Control may not be delegated to industry.
- o Mix: Not allowable.

J. Security:

- o Government: Military personnel provide the most confidence, due to screening, controls and indoctrination penalties, and supervision at total military facilities. For the same reasons, civilian personnel rate higher under this attribute.
- o Industry: Original developers also rate high, especially if they have developed the original system. For support contractors, security is lessened because additional people know about the system functions, vulnerability, etc. Thus there is a better chance for leaks just due to numbers alone.
- o Mix: A mix could permit substitution of unclassified generic test data in lieu of classified data during modification efforts. Critical mission software could perhaps be modified only by government personnel or only by the original developer (already having the basic information).

K. Deployability: In the course of evaluating, testing, or distributing systems, it may be necessary for PDSS personnel to go to military environments, such as on-board a ship or aircraft or to a forward military base. Security, training, and physical safety must be considered. Qualifications to fly in a combat aircraft, for example, are quite rigorous.

- o Government: Government, especially military, personnel who are already appropriate for a given situation can be more easily obtained if the government involvement is high.
- o Industry: While industry representatives can and do work in these environments, having them do so is generally more difficult, and sometimes more costly (for example if additional training is involved), than using government personnel.
- o Mix: The use of mixed personnel may enable the use of government personnel in military environments, but only if the technical qualifications match the jobs to be done.

APPENDIX F
TAXONOMY OF MISSION-CRITICAL COMPUTER RESOURCE SOFTWARE

SYSTEM TAXONOMY HIERARCHY

Level 1 - Application Categories

- Weapon Systems
 - Tactical
 - Air
 - Sea
 - Surface
 - Subsurface
 - Ground
 - Mobile
 - Transportable
 - Fixed
 - Space
 - Strategic
 - Air
 - Sea
 - Surface
 - Subsurface
 - Missile

- Intelligence Systems
 - Tactical
 - Collection
 - Analysis
 - Fusion
 - Strategic
 - Collection
 - Analysis
 - Fusion

Crypto and National Security
(Listed for reference purposes only)

Command and Control (Including Multiple Weapon Systems)
Tactical

- Fire Control
- Maneuver Control
- Communications

Strategic

- Fire Control
- Maneuver Control
- Communications

Direct Support
Direct Mission Support

- Mission Planning/Preparation
- Data Acquisition/Transfer

Test Systems
 Automatic Test Equipment
 Test Program Sets
 System-Specific Test/Simulation Systems
Training Systems
 Maintenance Training Devices
 Operator/Crew Training Devices
 Computer-Based Instruction/Management (CBI/CBM)
 Weapon System Simulation
Software Support Systems (PDSS Resources)
 Development tools
 Integration and Test Tools
 Baseline Configuration Management Tools

Logistics
 Local
 ILS Support Systems
 Personnel/Administrative
 Global
 ILS Support Systems
 Personnel/Administrative

Level 2 - Software Categories

Application Software
 Online, Time-Critical
 Offline

Diagnostic Software
 Built-in Test (BIT)
 Online
 Offline

Control Software
 Real-Time Executives
 Process Control
 Computer-Aided Design/Manufacturing (CAD/CAM)

System-Specific Support Software
 Translators
 Custom Compilers/Assemblers

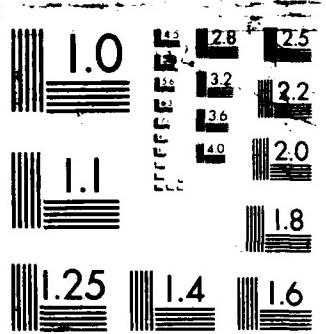
Vendor Software (Proprietary)
 Standard Operating Systems
 Compilers
 Utilities
 Data Base Management Systems

AD-A199 189 JOINT LOGISTICS COMMANDERS' WORKSHOP ON POST-DEPLOYMENT 2/6
SOFTWARE SUPPORT ((U)) JOINT POLICY COORDINATING GROUP
ON COMPUTER RESOURCE MANAGEMENT JUN 84

UNCLASSIFIED

F/G 12/3

ML



Level 3 - Implementation

Software

High-Order Language
Assembly Language
Microcode

Firmware

High-Order Language
Assembly Language
Microcode

APPENDIX G
ATTRIBUTES DRIVING WORKFORCE SELECTION

USER-ORIENTED ATTRIBUTES

U-1. Wartime Support

Definition: Ability of the workforce mix to sustain required PDSS activities in a period of global or localized hostilities/national emergencies.

Impact: Systems that are expected to have their mission-critical software substantially modified during hostilities (e.g., EW systems) must have a PDSS workforce that is reliable in such situations. This requirement is especially acute if PDSS support for such systems is located in a theater of operations. These system PDSS facilities must have their peacetime workforce structured with sufficient qualified military software personnel and a sustained personnel pipeline to ensure continuity of operations during hostilities.

U-2. Geographic Location (PDSS facility)

Definition: The location of the selected systems PDSS capability can be in a government CONUS facility, a CONUS contractor's facility, or a facility located in an overseas theater of operations.

U-3. Embedded Doctrine/Tactics

Definition: In many cases complex computerized weapon systems have established doctrine and tactics built into the system software. In effect, the established methodology for fighting the war is provided to the "user" through software, as is optimized system (and interfacing systems) performance.

Impact: A PDSS facility workforce must be formulated based on the desirability of having continuous support from the user community. The user personnel must be experienced in the doctrine and tactics of weapon system employment so that they can provide advice and guidance to PDSS professionals. Because many software changes that occur during PDSS can have a substantial impact upon established doctrine and tactics, the software designer must have access to sound user advice during the entire change process to avoid misdirection and costly mistakes.

U-4. Responsiveness

Definition: A PDSS concept for a system must be formulated with the requirement to be responsive to the "user" as one of the highest priorities. Accordingly, the workforce mix and staffing levels must be based on timely provision of the total spectrum of software services required by the user. It must be recognized that the user has unique requirements in a dynamic environment.

Impact: The PDSS staff must be based on identified tasks that involve not only software changes to programs, but also other services that may be required. To be completely responsive to user requirements, a PDSS staff might be formulated to provide:

- o Systems engineering support
- o Field service support
- o Training support
- o Interoperability support
- o Etc.

If an inadequate workforce mix does not allow for sufficient personnel with the required technical expertise, the PDSS provided to the user may be incomplete and lack in responsiveness.

U-5. Interoperability

Definition: Many systems have a requirement to interoperate with other systems to successfully accomplish their mission. Not only is the user of a system concerned with the operation of a specific system, but he is also vitally concerned with how his system operates as a member of a family of systems.

Impact: Interoperability integration and testing and periodic participation in special interoperability exercises of test beds may place significant burdens upon the staff. Unless the QA and test staff is adequately constituted, critical test activities cannot be properly supported by the PDSS facility.

U-6. Field Support

Definition: Software field support services include such tasks as technical assistance visits, troubleshooting assistance, training assistance, new software version delivery, support of tests and joint exercises, and providing continuous software technical expertise to the user.

Impact : To maximize effectiveness, software field service support services must be an extension of the professional staff of the PDSS center. This need arises because of the detailed knowledge that must be available to accomplish software field support tasks. The PDSS staff should be constructed based on the expected commitment to support the user's unique field requirements, in addition to fulfilling the more traditional role of the PDSS center. If this is not accomplished, valuable programmer/engineering resources may be periodically diverted to the field support role.

U-7. Training PDSS Personnel

Definition: This is the activity required to provide initial and continuing training to PDSS personnel.

Impact: The purpose of training is to develop employee skills and knowledge in order to increase proficiency in current or projected assignments. Training is an integral part of mission accomplishment that must be planned and budgeted. The initial training of PDSS staff for system software support should include formal and on-the-job training. The training should be in concert with the incremental assumption of system software into the PDSS activity.

Formal courses can provide additional language and technology skills for computer professionals on the current PDSS staff and for personnel with system experience transitioning to the software field. A predeployment overlap with the software developer will provide on-the-job training for PDSS staff.

U-8. Criticality of System

Definition: A critical system is one that is essential to the combat mission of the using organization. Functional inadequacy of a critical system's software may cause a serious degradation of operational capabilities or even non-operational status.

Impact: The mission criticality of the system may govern the size/mix of workforce and location of the PDSS facility and may lead to redundant PDSS activities.

U-9. Implementation Media

Definition: Software changes are usually incorporated into an updated baseline configuration, then distributed to the user in the field. The media for implementation of these changes can be tapes, disks, or firmware.

Impact: Some high-density software-intensive systems are employed throughout the world with US Forces. Implementation responsibility for frequent software changes to these systems can be a significant drain upon PDSS facility resources; this is especially true if firmware is the medium of change. PDSS workforce structures must make allowances for mobile implementation teams to deliver and install new software versions to fielded systems. Planning must also allow for the ancillary functions these teams are often required to perform, i.e., new software version training, staff briefings, verification, etc.

U-10. Change Process

Definition: The change order process is a key element of configuration management and is the machinery for incorporating changes into the baseline software. The user, via trouble reports, is usually instrumental in the establishment of the priority assigned to the changes.

Impact: The PDSS workforce mix must include a representative that can speak for the user community in matters concerning each change. This should be a military officer who has had practical experience with the system being supported or a similar system.

LOGISTICS-ORIENTED ATTRIBUTES

L-1. Number of Unique Users

Definition: The number of unique users will determine the number of unique software configuration baselines. Also, each unique user will probably have unique support requirements, i.e., tactics, geographical location, response requirements, etc.

L-2. Deployment

Definition: In addition to the environment (e.g., global, a sea environment, or a hostile environment), deployment includes locations and numbers of systems at each location.

Impact: Each environment presents unique support requirements. For example, a hostile environment may require a totally military support facility. Deployment to several locations may also require several software baselines, and these present special configuration management considerations. Deployment to allies may present additional support and configuration management problems. The deployment of a system may also require special change dissemination considerations.

L-3. Length of Life Cycle

Definition: This is defined as the expected operational life of a system.

Impact: The expected operational life of a system will determine the expected volume of modifications to the software. The longer the expected life, the more modifications we can expect due to changing mission requirements. These modifications are really new design and therefore require the appropriate support personnel.

L-4. Ancillary Requirements

Definition:: This refers to such things as computer operations, the requirement for software libraries, and hardware maintenance.

Impact: Failure to consider ancillary requirements means that the total support requirements will not be defined and the proper resources (personnel and equipment) will not be identified. In addition, the proper equipment, configuration management tools, development tools, and rights to proprietary software may not be included in the procurement contract.

TECHNICALLY ORIENTED ATTRIBUTES

T-1. Configuration Management of Software

Definition:: This is the process of controlling versions of software elements and systems to support the integrity of the system.

Impact: Absence of effective CM at any time in the software life cycle inevitably leads to a degree of functional disarray. The impact can range from minor delays to a total loss of capability. Any self-disciplined trained person or agency should be able to perform the CM function. Normally the CM function is done by the agency responsible for the software in the current phase of its life cycle. With close interfacing of agencies, it could be done by a second agency.

T-2. Quality Assurance of Software

Definition: Quality assurance of software includes those technical and management methods needed to provide developer-independent assessments to responsible management that software quality requirements are being (and have been) met. Existing software QA standards include MIL-STD-52779A and MIL-STD-SDS (draft form).

Impact: On any significant software development project a lack of adequate software QA effort inevitably leads to an unacceptable combination of low quality, late delivery, and cost overrun. To the extent that the PDSS effort includes any software modifications, a corresponding software QA effort is required to maintain project quality goals. Due to its independent nature, QA can be performed by any available qualified agency. If done by the same agency responsible for development or modification, QA should have an independent management path to the responsible manager.

T-3. Software Test Bed

Definition: A software test bed is usually a specialized test facility which generally includes special hardware and software needed to test software in an environment as close as possible to that of the intended application. Effective test beds can range from fairly small and simple ones which might be easily duplicated to extremely large and complex ones which cannot be readily duplicated.

Impact: Test beds are important to almost all embedded software systems. They often form the environmental basis for providing functional qualification tests needed to accept the software for end use. Effective use of a test bed generally requires serious participation by experienced personnel. The impact on the PDSS effort will generally require that the PDSS agency must use a test bed. The sophistication of the test bed may dictate the use of a PDSS agency.

T-4. Programming Language

Definition:: A given embedded weapon system will include one or more processors, each of which may execute instructions originally coded in one or more programming languages, such as Ada, Fortran, assembler, etc.

Impact: Not all computer programmers are equally proficient in the major programming languages. During routine maintenance, programmer proficiency may be significantly enhanced through good detailed design documentation. In PDSS strong consideration must be given to the number of available programmers having a sufficiently high level of language proficiency in the language(s) in use and to the available detailed design quality.

T-5. Proprietary Software

Definition: Proprietary software is software developed and owned by a commercial supplier. It is generally maintained by this developer and either not made accessible to others or only provided in some restricted manner (such as lease) in a form which practically prohibits any modification by the user.

Impact: If there is proprietary software included in a given system, some provision may be required for its continued maintenance by its original developer. Extensive interaction between the proprietary software and other software in the system may practically dictate the use of the proprietary software developer for maintenance of the complete system.

T-6. Adequacy of Technical Documentation

Definition: Acceptable minimum documentation standards are defined by various military standards (MIL-STD-SDS, MIL-S-1679A, MIL-STD-490, MIL-STD-1644).

Impact: Lack of adequate documentation inevitably leads to increased life cycle costs and reduced levels of reliability and maintainability. Documentation is usually the first part of a developer's responsibility which suffers from (unbearable) budget and schedule pressures. Such "compromises" simply postpone costs and propagate delays (in an amplified form) into the maintenance phase. Well-documented software can be far more easily maintained by an agency other than the developer than can inadequately documented software.

T-7. System Complexity

Definition: The man-machine interfaces, number of computers, networking, parallelism, architecture, number of lines of code, timing, computer loading, environment, and many other factors affect the complexity of the system.

Impact: The more complex the system, the more critical the PDSS mix decision. For example, in the aircraft area the addition of new missiles, displays, electronic equipment, etc., increases system integration requirements and thus mandates use of more skilled manpower to implement software support activities.

T-8. Software Engineering

Definition: The use of modern techniques such as modular programming, design-for-change approach, disciplined configuration management, etc., constitutes the state of the art of software engineering.

Impact: The use of modern software engineering techniques makes software less personality dependent, more understandable, and ultimately easier to enhance or support. It impacts the ability to complete contract efforts and reduces support costs. Good design and documentation reduce the skill level requirement of the support work force.

T-9. Software Maturity

Definition: This is the degree of experience achieved with the software code and logic, in terms of operational use and low rate of discovery of latent defects.

Impact: There is an inverse relationship between software maturity and support difficulty. Systems that have established logic and math flows, have well-known requirements, and/or are next-generation follow-ons to existing embedded systems tend to have fewer unknowns and thus become easier to support. A mature software system can be supported, on a relative basis, with fewer personnel and therefore is less costly or risky to support.

T-10. Technical Risk

Definition: The technical risk involves the number of unknowns in the system (tools, skills, experience, technology, complexity, state-of-the-art applications).

Impact: The greater the technical risk or the more unknowns in a system, the greater the requirement for specialized skills.

T-11. Built-in Test (BIT)

Definition: BIT is inherent software, firmware, or hardware logic which checks the validity and integrity of processing flow and output.

Impact: The effective use of BIT can result in significant improvements in system and subsystem reliability. However, BIT also adds another dimension to the complexity of the support process. Systems with extensive use of BIT will require specialized personnel and facilities to support the BIT implementation in the system.

T-12. Adaptability (Reprogramming Ability)

Definition: The ease with which the configuration of an embedded system can be changed.

Impact: Systems which change configuration frequently will impose tremendous strain on support systems. Associated with frequency of change is the issue of responsiveness in making the change. Systems requiring frequent short-lead-time changes will involve the creation of extensive deployed networks to accommodate the changes. This will probably mandate support structures that are dispersed, organic, and highly flexible.

PERSONNEL AND RESOURCES-ORIENTED ATTRIBUTES

P-1: Special Facilities

Definition: Special facilities (weapon system support facility/integration support facility) include buildings, equipment, and

ranges required to develop, test, change, exercise, or integrate software products.

Impact: The existence of special facilities may be a determining factor as to where all or a portion of PDSS should be accomplished. A predetermined government/contractor staffing mix may be inherent in the special facility. Special facilities may be located at the development contractor's facility and may be required for system-level activities. To avoid costly duplication of the facility, OEM PDSS may be indicated. Other facilities may be located at government installations and be staffed by government or support contractor personnel with special knowledge or skills pertinent to the operation of the facility.

Special facilities are a constraint on government/industry staffing mix. Special facilities should be considered during PDSS planning, and PDSS should be considered during facility planning and acquisition of special facilities.

P-2. Personnel Turnover

Definition: This is the rate at which personnel become associated with and then leave a company, project, organization, or activity.

Impact: Computer and software personnel are in great demand and in short supply. Opportunities for advancement often exist outside the computer professional's current organization. The 1981 industry turnover rate for computer professionals was approximately 28%. Although the turnover rate for government employees in 1981 was less than half the industry rate, transfers within the government could drive that rate towards the industry rate within a project or activity.

The turnover rate is most significant in PDSS staffing mix when the development contractor is expected to provide PDSS. The likelihood of that contractor's maintaining the development staff is low, nullifying many perceived advantages. The turnover rate of government employees in the PDSS activity may currently be as high as in industry. However, this turnover rate may be lowered by assuring an adequate grade structure. Military personnel policies limiting the length of an assignment push the PDSS staffing mix away from military.

P-3. Availability of Qualified Personnel

Definition: This is the number and location of personnel possessing computer, engineering software, and/or system-specific knowledge and skills available for assignment to a PDSS effort.

Impact on PDSS: Technical requirements should drive the PDSS staffing mix, and the mix should be made available through proper planning and implementation. PDSS is, and will continue in the near future to be, a labor-intensive activity. The availability of that qualified labor force is a determining factor in how a PDSS effort will be staffed. Currently all services are building PDSS staffs with the required computer skills. To ensure that system-specific knowledge is acquired by that staff, key PDSS personnel should be involved in the system development.

PDSS staffs consist of both government and contractor personnel. To ensure that the desired staff will be available for a system after deployment, proper and timely funding and staffing requests must be initiated early in the system development.

ADMINISTRATIVE/POLITICALLY ORIENTED ATTRIBUTES

A-1. Funding

Definition: Money provided for contracting support is designated by Congress under public law in specialized categories, i.e., production, operational, and maintenance and development. These are restricted to being applied within a weapon system category. Multiple types of money are required for support of some systems at different levels of need.

Impact: Decisions as to contractor/government support levels at PDSS are directly affected by the kinds and quantity of funds provided in a timely manner. Unless funding is matched against the planning documents, technical decisions will be made by administrative fiat.

A-2. Directed Procurement

Definition: Because of a number of factors (i.e., contractor technical expertise, proprietary data, test facilities,) it is cost-effective and timely to sole-source to a particular vendor. The statement of work can be somewhat general because the requirements are fully understood by the vendor.

Impact: Sole-sourcing greatly reduces the flexibility in decision making as to mix of personnel. The major problem is the availability of required funds for this manpower approach.

A-3. Competitive Procurement

Definition: In an attempt to reduce costs and comply with the strong emphasis for competition, the number of competitive contracts is being increased. A competitive contract statement of

work needs to be very detailed; therefore, a high level of expertise of government personnel is required for development and monitoring.

Impact: At minimum, a staff of experienced government personnel must be planned for when this type of contract support is utilized. This contracting may be viewed more as a level-of-effort extension rather than full support.

A-4. Personnel Ceilings

Definition: Most government agencies have restrictions on the number of civilian employees, their grade levels, and job series. The military has equal constraints on billets and number of qualified personnel.

Impact: Planning for government personnel in the mix may be unrealistic if the required positions in actuality cannot be filled.

A-5. Traditional Roles and Missions

Definition: Within the services, organizations want to maintain or obtain control of software, not because of time-effectiveness but because of perceived response to changed requirements, existing structure, or glamour of new technology.

Impact: These purely political decisions preclude effective planning of manpower utilization.

A-6. International Support

Definition: The services must provide support to Foreign Military Sales (FMS) and other programs that supply military stores to friendly countries.

Impact: These positions are included in the manpower ceiling of the military services, thereby competing for the technical manpower needed for US programs. Government personnel must be provided as a mix for all of these programs.

A-7. Security

Definition: Certain systems contain software which is sensitive in its nature, application, and/or origin. The sensitivity can be so intense that knowledge must be limited to a select few people.

Impact: The ramifications of security upon the government/contractor manpower mix are proportional to the decision about limiting the need to know. If, for example, an OEM completes a system and its associated software, the maximum limit of need to know could be

to the OEM and the cognizant service authority. Many "black world" projects are controlled in this manner, with the OEM providing software support. Conversely, software support for battle-area-deployed systems may be 100% supported by the government due to (1) limited contractor presence or (2) desirability for service-only support. In summary, the ramifications of security upon the manpower mix for supporting a system depend upon considerations of need to know, cost-effectiveness, availability of personnel, management decisions, and battle area proximity.

A-8. Acquisition Management

Definition: Implementation of acquisition activities can influence software support manpower mix dependent upon sequence, timeliness, deliverable items, adequate funding, and schedule.

It is possible to constrain an acquisition process to such a compressed schedule or meager budget that an unsupportable software product results. If one assumes that adequate funding and schedules are available, then the sequence of acquisition activities may indicate a manpower mix. For example, if a support facility is required for software support, then the facility may be developed in parallel with the software product. The facility turnover to the government might enable government support of the acquired software. Software cannot be supported without adequate engineering and technical data. If the timeliness of data delivery is not commensurate with acquired software, then the government could not provide support until such time as the data are available. In certain instances the quality of deliverables (software and data) prohibits easy software support.

Impact: Adequate acquisition planning and management must be enforced to assure that software is delivered with adequate descriptions and resources to provide support at the desired manpower mix.

A-9. Continuity of Operations

Definition: This term applies to the extent that failure-free software must operate. Factors involved are the extent of redundancy, degradation for component or software failure, and the strategic/tactical urgency for an operative system.

The requirement for continuous operation means that if a failure occurs it is detected and corrected or some substitute capability fills in the deficiency. Extremes go from self-correcting software to various levels of redundancy (or backup) to real-time failure fixes.

Impact: PDSS manpower mix depends upon the degree to which one strives for continuity of operation and the concept to provide the

continuity. Systems requiring high continuity of operation are usually associated with space or forward battle areas. In each case the manpower mix will vary widely.

A-10. Commonality of Applications

Definition: This is the extent of commonality that specific software has from system to system.

Impact: The main factor that would influence the total manpower for PDSS is the software complexity. For very simple software which applies to multiple systems, the PDSS manpower requirement is likely to be low and PDSS probably would be most cost-effective as a government-operated activity. As the complexity increases and as the multiple-system applicability narrows, the tendency is toward a more sophisticated, single support capability. The mix of manpower would be dependent upon a government decision as to the extent of contractor involvement.

A-11. Use of IV&V

Definition: Independent verification and validation (IV&V) is more effectively applied during full-scale development. Effectiveness is gained simply because it is cheaper to implement a software change as early as possible in the design/development phase. If discoveries of deficiencies are made in the later stages of software development, it may or may not be cost-effective to implement changes. On the other hand, noncomplex software does not warrant full-scale IV&V because it is not cost-effective.

Impact: A decision should be made before project development as to whether to use IV&V. Using IV&V has a tendency to reduce overall PDSS manpower requirements for a system but appears to have little effect upon the actual mix. The mix is simply a matter of where the government wants PDSS accomplished: contractor or government.

ORLANDO I

FINAL REPORT

PANEL B

INDEPENDENT VERIFICATION AND VALIDATION (IV&V)

MARCH 19, 1984

CO-CHAIRMAN:

CDR D. (Dave) Southworth
HQ, Naval Material Command (MAT 08Y)
Washington, DC 20360
(202) 692-3966

CO-CHAIRMAN:

John W. Sapp
Software A&E, Inc.
1401 Wilson blvd., Suite 1220
Arlington, VA 22209
(703) 276-7910

TABLE OF CONTENTS

	<u>Page</u>
4.2 Panel B - Post Deployment Software Support (PDSS) Independent Verification and Validation (IV&V)	
4.2.1 Objectives	4-2-1
4.2.2 Scope	4-2-1
4.2.3 Approach	4-2-2
4.2.3.1 Preparation	4-2-2
4.2.3.2 Panel Organization and Issues	4-2-2
4.2.3.3 Panel Operation	4-2-3
4.2.4 Discussion	4-2-7
4.2.4.1 The Effectiveness of IV&V Subpanel Report	4-2-7
4.2.4.2 Criteria for the Use of IV&V Subpanel Report	4-2-12
4.2.4.3 IV&V and the Software Life Cycle Subpanel Report	4-2-15
4.2.4.4 IV&V in the Organization Subpanel Report	4-2-22
4.2.5 Recommendations	4-2-34
Appendix A - Participants	4-2-A-1
Appendix B - Bibliography	4-2-B-1
Appendix C - Panel Presentations	4-2-C-1
Appendix D - IV&V Benefit Interdependencies	4-2-D-1
Appendix E - IV&V Experience	4-2-E-1
Appendix F - IV&V Case Study	4-2-F-1
Appendix G - Model Descriptions	4-2-G-1
Appendix H - Quantitative IV&V Cost/Benefit Analysis	4-2-H-1
Appendix I - Proposed JLC Policy Statement Concerning IV&V	4-2-I-1
Appendix J - IV&V Criteria: Supporting Material	4-2-J-1
Appendix K - Life Cycle Chart	4-2-K-1

FIGURES

	Page
4.2-1 Computer Software Development Cycle	4-2-16
4.2-2 Program Manager - Model 1	4-2-27
4.2-3 IV&V Entity - Model 2	4-2-30
4.2-4 Relation Between IV&V Levels and the Software Life Cycle	4-2-35
4.2-K-1 IV&V and the Software Life Cycle	4-2-K-6

TABLES

4.2-K-A IV&V Data Delivery	K-4, K-5
--------------------------------------	----------

4.2.1. OBJECTIVES:

Over the past decade there has been a dramatic increase in the number of planned and deployed Mission Critical Computer Systems (MCCS). These are systems which are of significant importance and which are integral to the effectiveness of today's military combat and support systems. This includes airborne, fixed and mobile ground, surface and sub-surface naval, and space systems which are required to operate in both hostile and benign environments. MCCS's are generally characterized as ruggedized programmable devices which exhibit high speed, accuracy and reliability in the processing and manipulation of data, performance of computations and in the exercising of system control. These features have and will continue to contribute to the development of military systems which meet or exceed performance, reliability and maintainability requirements and which demonstrate flexibility when responding to new requirements. MCCS's implement or aid in the implementation of systems and subsystem performance characteristics and serve to integrate the various elements into highly responsive and effective systems.

The embedded computer executes software. Thus, MCCS performance should be easily modified and/or enhanced by modifying (or replacing) the software. Normally software can be modified much faster and at a fraction of the cost of that which would be required to implement a comparable change in hardware.

The challenge to the panel has been to determine when and how much Independent Verification and Validation (IV&V) should be used in software development and in post deployment software support (PDSS). This is an extremely complex issue because of the overlapping roles of test, quality assurance, systems engineering and the service evaluation test departments. Too much checking is expensive and wasteful; too little is more wasteful yet.

What distinguishes IV&V from other areas, when to employ IV&V and the extent and amount of IV&V will vary as projects vary in size, complexity, criticality, staffing, acquisition mode and other factors. Very often, the project management is besieged with pressing problems. Any guidance to them on the use of IV&V should be as clear and definitive as possible.

The panel's objective was to sift through a wide range of positions, experiences, and issues to produce recommendations for changes and additions to JLC policies, procedures and standards with regard to IV&V. A secondary objective was to raise and define IV&V issues for further consideration which arose as part of the panel deliberations and which could not be adequately resolved during the workshop.

4.2.2. SCOPE:

The basic charter of the Independent Verification and Validation Panel was to recommend a Joint Logistic Commanders Policy that will clarify the use/non-use of IV&V in software development and in Post Deployment Software Support (PDSS). The policy should do the following:

- a. Cite the benefits vs costs of IV&V. This should include specific instances of benefits, quantified if possible.

- b. Specify general criteria for the use of IV&V. The criteria should consider, as a minimum, the following factors: types of systems, software applications, software complexity, type of acquisition and type of organization.
- c. Describe the use of IV&V during each of the phases of the life cycle including what products should be IV&V'd, such as documents, systems software, software modules, design, etc. Describe any specific IV&V issues in the development phase which will ensure the software is more supportable in the PDSS phase. Also included should be any "Rules of Thumb" for its use (e.g., percent of development cost for IV&V).
- d. Clarify the role of IV&V with software QA, test, systems engineering and other functions.
- e. Define organizational relationships to the IV&V process, who is responsible and what are the responsibilities.
- f. Identify areas where more study/data is required.

4.2.3. APPROACH:

4.2.3.1 Preparation

Prior to the workshop each of the panel members was sent packages containing background material pertinent to IV&V and to the panel's function. This material included:

- a. The IV&V Panel Charter which defined the objectives and scope of the panel and raised a number of significant issues to be addressed by the panel in the workshop.
- b. The Management Guide for Independent Verification and Validation (IV&V), August 1980, from Space Division, Directorate of Computer Resources, Los Angeles Air Force Station.
- c. Pertinent portions of Air Force Regulation 800-14 and AFSC Supplement 1 to AFR 800-14.

4.2.3.2 Panel Organization and Issues

The charter identified the following subpanels and allocated the known issues among them for their deliberation and resolution:

a. Subpanel 1 -- The Effectiveness of IV&V. Issues:

Is IV&V beneficial to the overall software life cycle cost? How does it affect reliability and maintainability? How does it affect ease of maintenance during PDSS?

What problems have been experienced with the use of IV&V? What are the lessons learned?

b. Subpanel 2 -- Criteria for the Use of IV&V. Issues:

What is IV&V? Independent? At what organizational level?
Verification? Validation?

How should the extent of use of IV&V depend upon the application,
complexity, software size, contract type, project office size?

What range of IV&V should be done for each of the following
during development and PDSS: user requirements, system
requirements, software requirements, algorithms, top-level
design, detailed design, code, test, documentation, etc.

What "Rules of Thumb" exist (percent of total cost for IV&V,
criteria)?

What are the criteria for the use of IV&V in development and in
PDSS?

c. Subpanel 3 -- IV&V and the Software Life Cycle. Issues:

How does IV&V fit into the software life cycle? At what specific
points is it employed? What is the relationship of IV&V to test,
integration, QA, system engineering, technical assistance,
software maintenance?

d. Subpanel 4 -- IV&V in the Organization. Issues:

Who should do IV&V? Can it be done "in-house"? Should PDSS
personnel be involved in development IV&V? Who should do IV&V of
PDSS software maintenance efforts?

What are the roles, relationships and responsibilities of the
project office, accreditation/test organizations, prime
contractor, corporate structure with respect to IV&V?

Each subpanel was also asked to address the question of what areas
require further study or data concerning the role of IV&V.

4.2.3.3 Panel Operation

The full IV&V panel held two sessions at the beginning of the workshop
to discuss administrative details and the method of operations for the panel.
At the conclusion of the first session, subpanel interest survey sheets were
collected from the panel members.

Based on the survey sheets the co-chairmen finalized subpanel
memberships and selected a chairman for each of the subpanels.

The second full panel meeting was held on Tuesday morning from 0800 to
1000. The subpanel assignments were announced, and a discussion ensued on
fundamental definitions of IV&V and the software development cycle.

The panel then divided into the four subpanels to meet in parallel sessions for the remainder of the workshop. The panel co-chairmen and subpanel chairmen met at the conclusion of each day to review and summarize progress. The results of the review activity provided cues for needed coordination among subpanels, as well as providing material for the daily briefing to the general membership of the workshop.

A final full panel session held Thursday afternoon from 1630 to 1730 provided a forum for each of the subpanel chairmen to report on the results achieved in his area. The results of these briefings and the comments they engendered were used as the basis for the summary briefing held Friday morning and for this report.

The following section summarizes the proceedings of the full panel session on 1 November and describes the Plan of Action and Milestones (POA&M) initially developed by each subpanel:

4.2.3.3.1 Summary of 1 Nov 83 AM Full Panel Session

Purpose: Obtain consensus on an acceptable definition of software development life cycle and of IV&V.

Approach:

1. Recommend use of the software development life cycle described in DOD-STD-SDS, and the definitions of IV&V from AFR 800-14, Vol I, AFSC Sup 1, 14 Dec 82.
2. Presentation and discussion by Dr E. R. Baker on MIL-STD-SQAM and the relation of IV&V to Software Quality Assessment and Measurement.
3. Presentation and discussion by Marshall Potter on available documentation and references concerning IV&V.

Issues and Resolutions:

1. There were no objections to the use of the MIL-STD-SDS definition of the software development life cycle by the panel.
2. A lengthy discussion of IV&V and SQAM concluded with an agreement to tentatively accept the AFR 800-14 definition of IV&V, but to consider, in the appropriate subpanel, definitions from JLC draft documentation and from the National Bureau of Standards.
3. A key issue raised during the discussion was: does the result of the panel deliberations -- recommendations for JLC policy, recommended changes to standards, and other documentation, etc., -- address existing military standards or should they address draft JLC standards which represent moving targets? It was felt that it is reasonable to target the panel's deliberations to the draft JLC standards.

4.2.3.3.2 Subpanel's POA&Ms

SUBPANEL 1 - THE EFFECTIVENESS OF IV&V

Tuesday

- Develop plan of attack
- Evaluate availability of data for quantitative considerations
- Evaluate quality of data (fidelity of prediction)

Wednesday

- Develop framework for data collection/modeling/sensitivity analysis
- Begin discussion of qualitative assessments

Thursday

- Complete qualitative assessments
- Examples of IV&V's success
- Wrap-up review; summary

SUBPANEL 2 - CRITERIA FOR THE USE OF IV&V

Tuesday

- Draft of Range of IV&V Activities

Wednesday

- Drafts of
 - Rules of Thumb
 - Extent of IV&V
 - Criteria for Using IV&V
 - Definition of IV&V
 - Additional Issues for Study

Thursday

- Final version of documentation on all areas above

SUBPANEL 3 - IV&V AND THE SOFTWARE LIFE CYCLE

Tuesday

How does IV&V fit into software life cycle?

- Define the total life cycle
- Identify the IV&V tasks
- Define the IV&V products

At what point is IV&V employed?

- Do we need IV&V in PDSS?
- Criteria for employment?

Wednesday

What is the relationship of IV&V to:

- Testing
- Integration
- Quality assurance
- System engineering (also software engineering?)
- Technical assistance (also education of managers)

Thursday

- Software maintenance
- Software maturation
- What are the areas for further study concerning the role of IV&V?

SUBPANEL 4 - IV&V AND THE ORGANIZATION

Tuesday

- Review issues
- Identify factors which affect the issues
- Begin development of organizational models

Wednesday

- Refine models
- Firm up set of factors
- Assess factors vs the models

Thursday

- Resolve differences in individual assessments
- Refine and organize conclusions
- Prepare final report

4.2.4 DISCUSSION

The following paragraphs summarize the subpanels' deliberations, discussions and conclusions.

4.2.4.1 EFFECTIVENESS OF IV&V (SUBPANEL 1)

1. Approach

The approach used by the subpanel was to identify generic costs and benefits; determine which of these could be quantitatively stated and conversely, those which could only be realistically stated in qualitative terms; determine if there were existing methodologies and models and data bases available to support a quantitative expression of the benefit/cost ratio; if so, reinforce the calculations with case studies and with a subjective expression of the costs and benefits; describe the interrelationship of IV&V with reliability, maintainability and other related factors to more precisely define cause-effect relationships, and provide a list of lessons learned to optimize the benefits of using IV&V.

The capstone question addressed was, "Is IV&V cost effective? If so, can it be expressed in terms to support the routine acceptance of its value?" Initial secondary questions revolved around the impact of IV&V on other programmatic areas such as reliability and maintainability. Derivative questions surfaced during panel deliberations. Those questions involved the issues of where directed use and supporting analytical models to support the direction were appropriate and/or feasible.

The primary goal was to reach consensus on whether IV&V is beneficial. Secondary goals were to determine if benefits and costs could be quantified and to interrelate the effects of IV&V to life cycle program sub-elements such as Reliability, Maintainability and Availability (RMA). Derived goals were to provide a starting point for more precise estimating tools for PMs to support IV&V activities and to develop a forceful but pragmatic policy statement on the requirement for IV&V in programs.

2. Issues Considered

- Is IV&V cost effective?
- Can benefits/costs be predicted?
- Can resourcing required for IV&V be protected?
- Does IV&V provide benefits/costs to the software system as a whole or only to selected portions of the system?

3. Findings

IV&V is beneficial and appears to be cost effective. Its impact is felt in potential improvement of the total software system subcomponents and activities throughout the life cycle. The cost benefit ratio can not be absolutely measured because the introduction of IV&V into a program changes the very nature of the systems under investigation resulting in no pure baseline for comparison. Databases and methodologies exist to parametrically estimate the cost/benefit ratio. The precision and confidence of the estimate is not known due to several factors. The databases were constructed some time

ago for purposes other than for quantifying IV&V benefits. Methodologies and models have not been validated in a controlled environment. Future work in data collection, modeling, and validation areas show great potential for improving the precision of the estimate. The resources required to do this work have not been estimated. A more detailed discussion is contained in Appendix D.

There are numerous substantial subjective benefits resulting from the use of IV&V. Professional judgment of panel members, gained from extensive experience both in government and industry, was used to evaluate the impact on the decision process to use or not use IV&V. The consensus was that, in many cases, these qualitative factors would carry as much or more weight as the quantitative factors. A more detailed discussion is contained in Appendix E. Although controlled experiments have not been conducted to deterministically predict IV&V benefits, actual use of IV&V on different programs provides some insight into the qualitative benefits of IV&V. A limited number of "case studies" were hastily analyzed to summarize a common perception in some documentation that IV&V is a beneficial processs. As in most complex systems, the cause-effect relationship of IV&V and system improvement cannot be absolutely determined, although, a pragmatic assessment of the "case studies" shows that a cause-effect relationship does exist. This summary is also included in Appendix F.

The quantitative methodology investigated by the panel has utility for predicting IV&V resourcing requirements. If the decision is made to apply IV&V, the program manager could use the model to determine funding requirements. There are additional heuristically based models which also have that capability. Some of the models and their descriptions are included in Appendix G.

4. Conclusions.

Independent Verification and Validation (IV&V) is an established means of analyzing software programs which has been shown to be beneficial to the development and entire life cycle quality of virtually all mission-critical computer systems software.

The benefits derived from IV&V are many. Some benefits can be measured and quantified in terms of time, dollars, or resources. Some benefits are, by necessity, unmeasurable and can be addressed only in qualitative terms. A summary of the benefits of IV&V during a development effort, based on the experience of the subpanel members, follows:

A. Quantifiable Benefits

- 1) Earlier detection of software errors, which reduces the impact of repair. The earlier software errors are discovered and identified the more adequately the correction time and resources can be planned for.
- 2) Program risk is reduced, both in terms of the development effort and of weapon performance. In the development effort, IV&V can directly influence schedule and cost by reducing schedule slippages and cost overruns.

8. Qualitative Benefits

- 1) An IV&V program raises the level of confidence for the PM and his entire staff.
- 2) The IV&V program provides an independent, second evaluation of key ingredients to the software development.
- 3) Better software documentation evolves during the development cycle, positively affecting both the development and cost to document.
- 4) With proper management the prime contractor, as well as government agencies, can work in cooperation with an educated IV&V entity to solve problems as soon as they arise. The IV&V team embodies a resident knowledge of the software that can equal that of the prime contractor.
- 5) IV&V supports the PM in management reviews by providing independent data and assessment.
- 6) The products of IV&V facilitate later CM tasks, such as FCA.
- 7) IV&V improves technical performance, schedule, cost, and CM visibility during all phases of the development.
- 8) IV&V provides data and advice on freezes, baselines, and cutoffs at major and minor milestones during the development.
- 9) IV&V can reduce the "black magic" aspect of the Project Office's understanding of the software development by being available to explain ideas, problems, and events in everyday language.
- 10) IV&V is a motivation to the prime contractor to do the best job possible. The presence of an IV&V contractor has demonstrably changed the prime contractor's methods in several instances.
- 11) In the case of multiple computer program configuration items (CPCIs) for one weapon system, each developed by a different subcontractor, the IV&V contractor often has the best knowledge of the total, integrated software and the entire set of program interfaces.
- 12) Error corrections are evaluated for accuracy. This amounts to closed-loop evaluation with its associated confidence.

In a post deployment software support (PDSS) environment, IV&V plays the same roles as above, as modifications or improvements are formulated and executed. In addition, IV&V before and during PDSS carries these benefits:

- 1) In the transition to PDSS, proper development IV&V guarantees a set of software documents that are complete and correct -- a must for timely and reasonably-priced PDSS.
- 2) PDSS IV&V facilitates a continuous operational capability in the face of modification or enhancement by reassuring correctness and providing insight into software changes.

C. Benefits vs Cost for IV&V

Where the benefits of IV&V can be quantified, many models have been proposed and used to show benefit in real dollars. An example of such a model is described in Appendix H.

Where the benefits of IV&V cannot be quantified, the PM or entity responsible for the software must evaluate each non-quantifiable benefit in light of his or her particular situation and asses how much benefit they expect to receive in terms of program stress, morale, and expectations. Obviously, these factors will vary from person to person and from project to project, but in all cases, the benefits should be considered to be over and above the resource savings gained from the quantifiable benefits.

D. Lessons Learned in IV&V

The group experience of subpanel members led to agreement on lessons learned. It is believed that these items can be of benefit to others involved in IV&V.

- 1) It is beneficial to begin IV&V as early as possible. Ideally it begins analysis tasks as soon as a system level specification is available.
- 2) Whether or not resources for IV&V are limited, the effort should be thoughtfully prioritized and tailored to the critical areas of the software.
- 3) It is very beneficial to incorporate provisions in the developer's contract or charter that facilitate IV&V. This requires involvement by an IV&V expert in the formulation of the prime contract.
- 4) The IV&V program (and its IV&V management plan) must be made and considered flexible, so that schedules and manloading can be made to follow and respond to the "real world" of the development program.
- 5) IV&V accomplishments and successes should be recognized and publicized.
- 6) Records should be kept of all IV&V analyses and findings, both formal and informal, to guarantee both IV&V performance tracking and data availability for future programs.
- 7) Both formal and informal change procedures need to be established early and maintained to provide timely and correct data flow to and from the IV&V team.
- 8) An IV&V manager should work to establish and carefully maintain a smooth, orderly, and diplomatic relationship between the IV&V contractor, the project office and the prime contractor.
- 9) The IV&V staff's skill and qualifications are a more critical ingredient than the IV&V tools used.

It is worthwhile to postulate a JLC policy statement (even though not supported by the subpanel charter). There is strong evidence that cost/benefit analysis is feasible, that implementation mechanisms exist (resourcing prediction) and the directive channels are available (DODDs, AFRs, etc.). The subpanel, therefore, formulated the policy statement in Appendix I. The main point made in the policy statement is that IV&V is as important as other parts of the development/PDSS activity and should be seriously considered.

5. Recommendations.

- Subpanel policy in Appendix I be issued in an IV&V policy directive.
- JLC endorse further data collection model improvement, model exercise and calibration activities to provide a more precise resource prediction capability to PMs for solicitation of resources for total program implementation and to decision makers to insure that programs have seriously considered the use of IV&V.

4.2.4.2 IV&V CRITERIA (SUBPANEL 2)

1. Approach

The IV&V Criteria subpanel was chartered to develop a definition of IV&V, define criteria for the use of IV&V, and develop an approach for identifying an appropriate range of IV&V activities. The subpanel considered alternative perceptions of IV&V, ranging from the type of IV&V typically associated with mission critical applications to a set of activities embracing all checks that might be performed to enhance software life cycle planning, reliability, and supportability. These disparate perceptions of IV&V became a key issue as they impact the definition of IV&V criteria, the role of IV&V in the life cycle, and organizational issues being addressed by other subpanels.

The subpanel adopted the IV&V definition contained in the JLC Policy on Computer Resource Management as the basis for further discussion. Next the subpanel examined the potential range of IV&V activities that might be performed during development and PDSS, including IV&V of

- o System Requirements
- o Software Requirements
- o Algorithms
- o Top-Level Design
- o Detailed Design
- o Code
- o Test
- o Documentation
- o Reviews/Audits

Each of these IV&V activities was further subdivided into constituent subactivities that would accomplish the corresponding IV&V activity.

The subpanel next determined the conditions under which these various IV&V subactivities would be appropriate. This analysis led to a consensus that a small set of discrete IV&V levels could be identified and that criteria could be developed to guide the Government project manager in determining the appropriate IV&V level for his project.

2. Findings and Conclusions

2.1 Definition of IV&V. The subpanel considered current JLC, Air Force, National Bureau of Standards (NBS) and other candidate definitions of IV&V. The most suitable definition of IV&V was found in the JLC Software Quality Program policy. However, this definition contained minor terminology inconsistencies and minor changes are recommended in paragraph 1 of Appendix J.

2.2 IV&V Levels. Four levels of IV&V were identified ranging from (for lack of more appropriate names) "bare bones" through "full blown" IV&V efforts. The subpanel recognized that early IV&V activities result in greatest payoff, therefore less intensive IV&V levels should focus resources on these high-payoff activities. The goals of each of the four IV&V levels, shown in paragraph 2 of Appendix J, were defined with this principle in mind.

2.3 IV&V Criteria. Criteria are needed to determine which level of IV&V is applicable to any given project. Although the AFSC Space Division "Management Guide to IV&V" provides an excellent foundation, it was found deficient in that it considers only a minimum number of factors. A more suitable concept was found in the JLC Software Quality Program Policy's criteria for determining SQAM independence (paragraph 3 of Appendix J).

The subpanel suggested that a numerical rating system serve as a means of determining the level of IV&V. This numerical rating system could be developed from the SQAM criteria, as follows:

- o Each criterion could be assigned a numerical weight, indicating its relative importance within the system
- o Each criterion could be assigned a numerical risk rating (replacing the current risk ratings of high, moderate, and low)
- o A score could be calculated as the sum of each criterion's weight multiplied by its risk rating

The total score would indicate which IV&V level should be employed, ranging from no IV&V to full blown IV&V. The criteria, weighting schemes, and risk ratings remain to be formulated.

The subpanel recommended that studies should be performed to augment the SQAM criteria as necessary and develop a numerical rating system. The subpanel further recommended the following modifications to JLC documents:

- o Software Quality Program Policy - add the augmented criteria, rating system for assigning IV&V levels, and definitions of IV&V levels in terms of their constituent subactivities.
- o MIL-STD-SQAM - Incorporate revised definition of IV&V and add IV&V task statements for each IV&V level.
- o Software Quality Program Guidebooks - add suggested assignment of tasks to IV&V levels in Volume II.

2.4 Range of IV&V Activities. Major IV&V activities were identified as System Requirements Verification, Software Requirements Verification, Algorithm Verification, Top Level Design Verification, Detailed Design Verification, Code Verification, Test Analysis/Validation Testing, Documentation Analysis, and Participation in Reviews and Audits. For each of these major activities, constituent subactivities were identified. Then, the subgroup determined the subactivities applicable to each IV&V level. The results of this analysis, shown in paragraph 4 and 5 of Appendix J, reflects the goals established for each IV&V level.

3. Recommendations

3.1 Definition of IV&V. Modify the definition of IV&V in the JLC Policies on Computer Resource Management and Software Quality Program in accordance with paragraph 1 of Appendix J.

3.2 IV&V Criteria. The following recommendations for further study were made:

a. The following factors should be studied for possible inclusion as criteria for the extent of IV&V.

- (1) Programmatic complexity (type of procurement, schedule constraints, etc.)
- (2) Environment for use (application, i.e., weapon system, ATE, etc.)
- (3) Overall cost (total program cost)
- (4) Hardware limitations (memory constraints or ability to rehost, etc.)
- (5) Program Office resources (funding, personnel, etc.)
- (6) Modifiability of original baseline (to be considered for enhancements or add-ons to an existing program)

b. In order to develop a numerical weighting system, schemes need to be formulated for:

- (1) Assigning numerical risk ratings (to replace "high, moderate, low")
- (2) Weighting each criteria to indicate its relative importance.
- (3) Calculating a numerical score which can be mapped into the appropriate IV&V level (ie. bare bones, low, moderate, full blown).

4.2.4.3 IV&V IN THE LIFE CYCLE (SUBPANEL 3)

The subpanel on IV&V and the life cycle considered ideas on how and where the IV&V tasks fit into the software life cycle. In addition, this subpanel described the relationship of IV&V to various facets of software development and PDSS. The method of work was to address ourselves to various questions and their component parts with the overall goal of producing a graphic representation of how and when IV&V fits into the software life cycle.

We considered several items:

- How does IV&V fit into the software life cycle?
- What is the total life cycle?
- What are the IV&V tasks?
- What are the IV&V products?
- At what point are the various IV&V tasks applied?
- Do we need IV&V in PDSS?
- What is the relationship of IV&V to testing, integration, quality assurance, system engineering, technical assistance, software maintenance, and software maturation.

We also considered areas in IV&V that require further study.

We concluded that the total life cycle was a continuing loop of the MIL-STD-SDS (draft) model for system and software development (Fig. 4.2-1). Post deployment software support includes in some degree or other, all of the same steps in that development cycle.

Figure 4-2-K-1 in appendix K is a graphic representation of how and where IV&V fits into the software life cycle, including PDSS. The following paragraphs further discuss and report our findings and conclusions to the previously mentioned questions.

While discussing IV&V in PDSS we discovered some additional areas that might require IV&V efforts since they are often encountered in PDSS but are not necessarily within the pre-deployment software life cycle. These areas are:

- An upgrade in hardware using the same system.
- An optimization of a compiler.
- Removing patches and incorporating them into the source code.
- Optimizing the source code (removing dead code, un-needed routines short of a redesign).
- Checks for the actual system performance boundary.
- The education of program management staffs and other IV&V contractors for continuity.

Findings and Conclusions

The following series of statements represent the findings and conclusions of subpanel 3. They are not purely a single set of conclusions derived from the above discussion but are a series of observations generated by subpanel members during discussions on the assigned topic. They are not tightly restricted in subject matter, nor are they in total agreement. They do however, offer specific insight into the application of IV&V across the life cycle of a system that incorporates software.

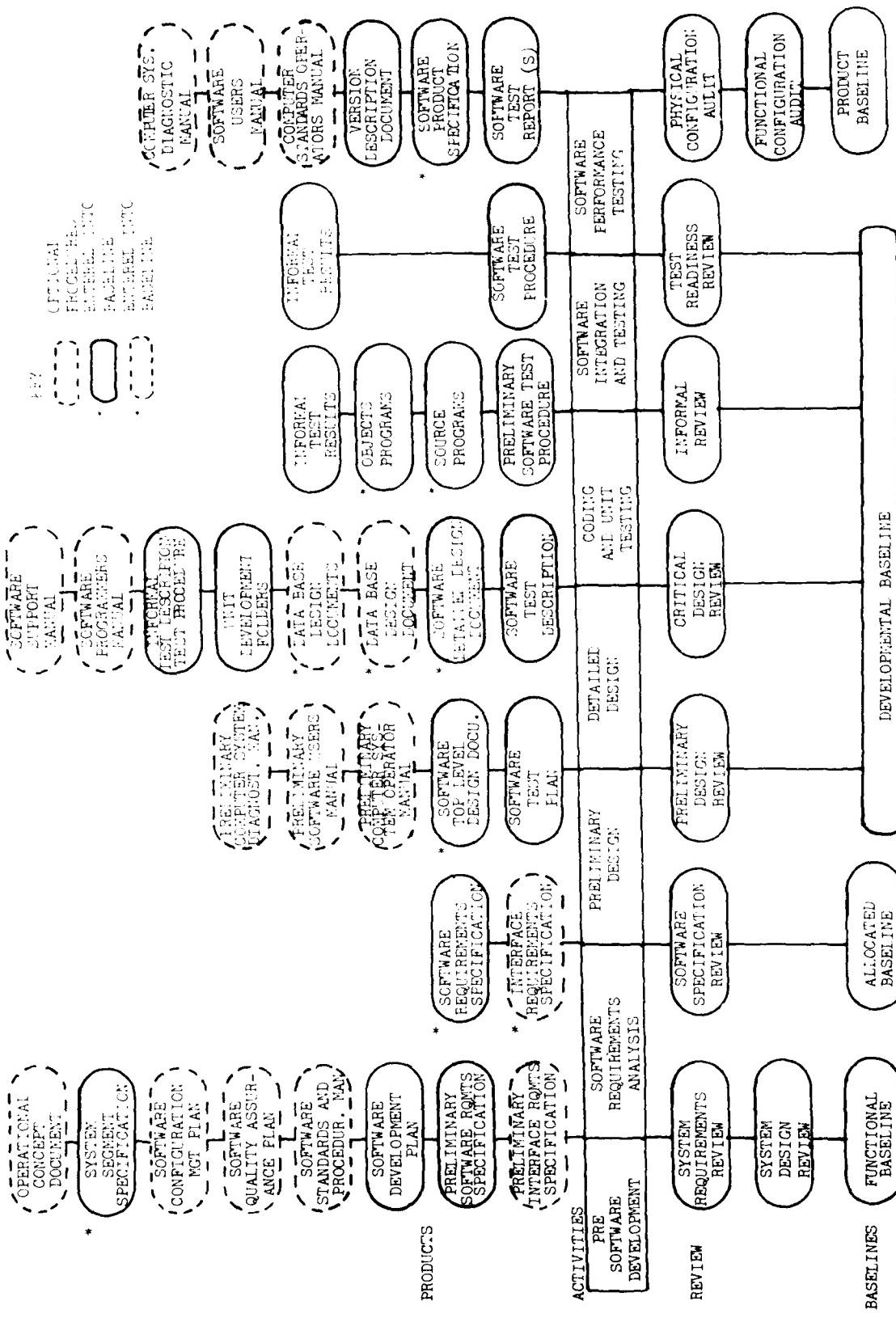


Figure 4.2-1 COMPUTER SOFTWARE DEVELOPMENT CYCLE

1. IV&V is difficult, labor intensive, expensive, and impacts program schedules.
2. It is better to do IV&V than not with respect to mission-critical systems.
3. Decision-making with regard to development of mission-critical systems tends to be driven by schedules rather than by best technical approach.
4. As a consequence of finding 3, the need for IV&V increases as confidence in the product decreases due to schedule-driven compromises in approach.
5. The cost of post deployment support can be reduced by adequate IV&V prior to deployment.
6. However, since mission-critical systems characteristically have a long life cycle with continuing system modification in response to changing mission requirements, "development" never really ends. Thus, the pre-deployment life cycle activities are all continued after deployment of an initial version of the system.
7. A consequence of 6 is that pre-deployment IV&V activities are also appropriate during PDSS.
8. There is a potential for supplementary IV&V activities required for PDSS. Examples include:
 - Verification of the compatibility of the software engineering environment used during development with that used for post-development support (i.e., will the delivered baseline be maintainable using the standard tools available to the post deployment software support activity?)
 - Code retirement validation (i.e., assuring that exactly all of the code required to be retired from a deployed system is, in fact, excised).
 - Verification of patch replacement (i.e., does the executable software built from revised source code perform the same as the previous, patched program? Also, has appropriate documentation revision been performed?)
 - Verification that factors affecting post-development support are addressed during development (for example, verifying that machine-dependent and operating system-dependent functions in the software are minimized and encapsulated so that the post-deployment introduction of improved standard processors will be facilitated).
9. Automated support tools for IV&V should be incorporated in software engineering environments under development within DCD.
10. IV&V is applicable not just to deployed software, but also to simulation/stimulation software, test scenario generators, and other software products that directly support the development and maintenance of the deployed product.

11. IV&V data produced during development of the application product must become part of the product baseline. Such data provides historical perspective for post-deployment software support. Further, availability of such data is a valuable asset for test, integration, quality assurance, system engineering, and software engineering teams involved with product development.

12. One of the best ways to train personnel for post-deployment software support is to involve them in IV&V activities.

13. IV&V must include checks on CM to ensure adherence to CM and library control procedures.

14. There is a need for IV&V in the maintenance environment (PDSS). There is also a need for IV&V during the development phase to ensure the software product is maintainable using a standard maintenance facility.

15. Confusion exists between the software development process and the software life cycle.

The process is the translation of requirements into code.

The life cycle consists of serial iterations of the process. With each iteration some portion of the previously developed software is carried forward and integrated into the new version. Post Deployment Software Support is characterized by the fact that each version consists largely of software carried forward from the previous version.

16. The development process (which repeats and repeats throughout the life cycle) always requires some form of verification and validation.

17. Experience has demonstrated that independence is an essential characteristic of successful verification and validation. It is not effective or fair to ask or allow the individual or group who develops a product to evaluate that product.

18. The allocation of verification and validation tasks to government agencies or contractor resources is primarily a question of availability of qualified personnel.

19. Few software developments actually begin from scratch. Most use some previously existing software, if only a compiler and an operating system (OS). The tendency in the future, as software becomes more and more reusable will be that "new" developments incorporate more "off-the-shelf" code. Thus, the differences between post deployment software development and pre-deployment software development will decrease.

Software development and software management involve a wide range of disciplines. System engineering, systems analysis, software engineering, configuration management, test engineering and quality assurance are all involved in producing a successful software product.

The developing contractor employs all of these disciplines in a development role, the acquiring project manager employs personnel from the disciplines in a management role. In addition, he may utilize resources in a third role -- independent verification and validation.

A possible analogy that clarifies the relationship of the IV&V effort to the functions listed above is that of a doctor's patient, who, faced with a recommendation for major surgery, calls in another doctor and asks for a second opinion. That is, an independent analysis of symptoms, test results, and procedures aiming at an independent conclusion that either supports or challenges the original recommendation. Given competency on the part of both doctors, the second opinion will either provide the patient with the confidence to proceed with the operation or alternative choices to be considered. The IV&V activity, like the second doctor, is hired to provide an expert opinion on the status of the software development process. Unlike the second doctor, the IV&V agency generally operates in a continuous manner, providing an ongoing assessment of the software development process. The IV&V agency forms its opinion by applying the methodologies of software engineering in a cost effective manner. Thus it uses a mix of system engineering methods, quality assurance methods, test and evaluation methods, and configuration management methods as necessary in addition to assessing the results of such activities already performed. The degree to which the IV&V agency repeats or duplicates the activities already performed under such disciplines, depends on the criticality of the software in question and the adequacy of the data developed. Shortages of resources, however, have often resulted in the IV&V agency being the principal evaluator or collector of data in these areas. IV&V must be tailored to be cost effective. Therefore, some division of tasking between contractor resources, project manager resources and IV&V resources will always occur. This does not mean, however, that there exists some natural division between QA and IV&V or that there is some clear distinction between IV&V and test and evaluation. The disciplines of systems engineering, software design engineering, quality assurance, configuration management and test engineering are all necessary for successful software development. Likewise the IV&V agency must consider and evaluate all of these in forming an independent assessment of the status and condition of the software product under development and in generating recommendations for actions by the program manager.

Relationship of IV&V to Test, Integration, and QA

IV&V activities during software development and software maintenance encompass similar activities to those of test, integration, and QA; the primary difference is in the independence and the degree of involvement based on criticality of the software.

For testing, both QA and IV&V are involved in reviewing and evaluating test plans and procedures. QA activities are primarily for verifying completeness and conformance to standards while IV&V is more concerned with verifying adequacy and traceability of requirements. The involvements of IV&V becomes more intensive as the testing progresses from unit test to CSCI validation, integration, and system level testing; QA activities during test conduct involve monitoring and witnessing testing for compliance with procedures while IV&V is concerned with evaluating adequacy of test cases and test results. In addition, IV&V may conduct tests for critical software of critical functions. IV&V also supports government agency testing by assisting in preparation of test procedures and test conduct as required.

QA and IV&V of the software development and maintenance process are already related; many of the activities are similar in nature but differ in intent. Review of software documentation by QA is focused on completeness and compliance to standards and procedures while review by IV&V is focused on adequacy, traceability of requirements, and identification of deficiencies from a user/mission point of view.

QA is also focused on auditing the development process for compliance to approved plans, such as the software development plan and the software CM plan. IV&V is focused on the adequacy of such plans.

(Note: A minority opinion held that the above discussion implied a definition of QA, while the original JLC approach was to define appropriate functions required to evaluate the software product, associated documentation, and processes, leaving the allocation of these functions to the commands or program managers. These evaluation functions, including both assessment and measurement tasks, encompass what IV&V, Test & QA organizations are thought to perform.)

Areas Where Further Study is Required

1. Should "reused" or "retained" units of software be treated differently for IV&V? If so, how? To what extent does software reutilization mitigate against the need for IV&V?
2. What set of IV&V tools are appropriate to build into emerging standard software engineering environments? Are different tools required according to whether the SEE has an orientation to Ada or to some other languages?
3. If a standard set of tools is not identified or developed, what criteria should be used for selecting from among alternative offerings of IV&V tools?
4. How does the expansion of the use of microprocessors and firmware affect IV&V methodology?
5. How should IV&V of distributed processing systems differ from that of centralized applications?
6. What contractual mechanisms will assist in assuring adequate IV&V is performed? Do these differ if the application is under development by a prime contractor and several sub-contractors?
7. What documentation is necessary for the IV&V deliverables in order to put them to their proper use?
8. What IV&V procedures are required to deal with code optimization options (e.g., use of optimizing compilers, suppression of run-time checks, etc.)?
9. What IV&V procedures are appropriate for determining, during post deployment software support, that the accumulation of software modifications has consumed the original system design and that it would be better to redesign the whole system than to further modify the deployed software?

10. What, if any, additional formal reviews should be established as part of post deployment software support?

11. What specific IV&V functions are required for security requirements of software systems?

Recommendations

1. That the JLC publish the information in Appendix K in a policy document.
2. That the issues raised under areas of IV&V that require further study be considered for the next JLC software workshop.

4.2.4.4 IV&V IN THE ORGANIZATION (SUBPANEL 4)

INTRODUCTION

1. Subpanel Objectives/Products

Identify issues and alternatives as guides to DOD commanders and program managers addressing allocation of V&V or IV&V tasks to their organizations or contractors for software development and PDSS.

2. Questions to be addressed

- a. Who should do IV&V? Can it be done "in-house"? Should PDSS personnel be involved in development IV&V? Who should do IV&V of PDSS software maintenance efforts?
- b. What are the roles, relationships and responsibilities of the project office, accreditation/test organizations, prime contractor, and corporate structure with respect to IV&V?
- c. What are the areas that require further study/data concerning the role of IV&V?

The questions regarding IV&V and its allocation to organizations are extremely complex and depend upon numerous factors, some of which are outside the control of the Program Manager.

One of the factors is the extent of V&V tasks or functions. For example, IV&V could include work efforts by the developer, program manager, product assurance, testers and users depending upon the definition of IV&V. Thus, allocation of IV&V tasks to particular organizations, particular projects, particular commands, or particular services depends upon that definition.

Another factor affecting allocation of V&V tasks is the regulatory constraints placed upon the project, command or service. Such constraints constitute the "business" structure of that entity and may dictate which organizations perform V&V.

Interrelated to the definition of V&V and the business structure is the meaning of independence, for interpretation of "independence" may determine who is assigned V&V responsibilities within a specific project, command and service.

For those and other reasons it is difficult to mandate a general, joint service policy as to which specific organizations should perform IV&V. Instead, only issues which should be considered by a service in deciding allocation of IV&V tasks will be addressed. Secondly, management models will be described and evaluated against the issues in terms of advantages and disadvantages for each management model.

DEFINITIONS

The definition of elements of IV&V are, as discussed earlier, critical factors for the allocation of IV&V tasks to organizations within the software development community. Under the philosophy proposed by the Joint Logistics Commanders, verification and validation is a set of tasks whose ultimate goal is to evaluate, assess and measure (test) the software development process, products, and associated documentation and provide feedback of the evaluation to PM, commander, or service as appropriate. (Here the word "development" as used for software includes the development in the PDSS environment.)

Assignment of the evaluation tasks (IV&V) to organizations is left to the individual service implementation.

The definitions used by this subpanel were those developed by subpanel 2 (see Appendix J).

ISSUES FOR COMPARISON

The IV&V subpanel discussing IV&V in the Organization decided to investigate a limited number of organizational models and compare them against each other by using a list of issues. Six major groups of issues were initially agreed upon by the members of the subpanel. These issues, capability, resources, time, criticality, side benefits, and independence were not understood by any members of the subpanel to be orthogonal to each other. There are several interrelationships among the different issues. These interrelationships are discussed in some detail.

The issue of capability of the IV&V group to accomplish their mission was subdivided into two subcategories; (1) Functional Capability and (2) Training and Skills. IV&V is a process where an independent group of computer system professionals are assigned the responsibility to provide a continuous oversight to assess, review, analyze, test and verify correct system performance of the software. Capability of the group is divided into functional capability to use the facilities and tools available in an effective manner and the overall capability of the IV&V group to assess the problem space effectively. Two separate backgrounds are needed by the IV&V professional. One is an understanding of the specific problem space (e.g. command and control, communications, navigation, intelligence gathering, air traffic control etc.) and the other is associated with the tools and methods of software IV&V. In both cases it is understood that the IV&V group will have to provide training to keep the skills and talents of the established, as well as new members of the IV&V organization, up to a level of excellence so that they can effectively perform the IV&V mission.

The issue of resources was divided into five subcategories. The first was one of funding. The perspective the subgroup took was, if the function of IV&V was required, what organization would be able to accomplish the mission successfully with the minimum amount of financial resources. Some members of the group expressed concern that management would potentially use an IV&V organization that was equipped to do only a minimal amount of IV&V. This concern was relieved by the fact that another subpanel was addressing the degree of IV&V required and had made models for IV&V from "bare bones" to "full blown" and was investigating when, where and how to apply these different levels of IV&V.

The next resource addressed was people and skills. The following points were discussed. Do the people with the requisite skills exist in sufficient quantity today to meet our needs? If people are available, would they lose their skills and talents in some organizations faster than in others (e.g. Headquarters versus a field activity or service laboratory)? Would the IV&V organization be able to make maximum use of the skills mix within the overall organization better than another organization (this is related to the concept of "economy of scale")? Does one organization have a better "environment" that would help recruit and hold IV&V personnel?

Under the resource subcategory of facilities the subpanel looked at two perspectives, one was how close the organization was geographically to both the PM and the Developer (i.e. location). The other perspective looked at the quality of the facilities in order to attract quality personnel and make the work enjoyable.

The subcategory of resources relating to tools also had two perspectives. Are the tools adequate to accomplish the mission? This is important in that new tools can not be easily integrated into the organization if experience on their effectiveness is not already in place. An additional perspective addressed the capability and potential costs of acquiring the data rights to all IV&V tools.

The final Resource subcategory was security. In the main, this issue addressed the capability of an IV&V group to acquire a sufficient number of cleared billets to accomplish a job and to secure the work environment from unauthorized access. Most Mission Critical Systems (MCS) are of sufficient importance to DOD to require some concern for security, even if no classified information is involved. Some MCS deal with extremely sensitive classified information that is very tightly controlled on a need to know basis. In all cases the effectiveness of the IV&V organizations' capabilities in regard to security have to be addressed.

The third major issue was Time. Time is associated with schedule. Two important points were addressed by the panel. One was, how long it would take to get an IV&V group up and working. The other perspective was, what effect would the different prototype IV&V organizations proposed have upon the schedule if there were problems or difficulties involved in the delivered projects? It should be noted at this point that one organization may be easy to get started yet have a high negative impact on schedule. As a result of this internal interdependency, the views of the members of the subpanel were somewhat divergent.

The issue of Criticality addressed the effectiveness of a particular organization upon the most critical DOD systems. Is one organization more "robust" in meeting its mission if a highly important and critical system is placed under its authority? Another pertinent question addressed the fact that a critical system may require extensive IV&V. Is one organization better equipped to accomplish this task than another?

The subpanel addressed the issue of Side Benefits (effects) in some unique ways. The most obvious side benefit was a potential training of the PDSS organization in advance of its mission of maintaining the software. Other side benefits such as corporate memory, enhancement of the IV&V tool set and

effective on-the-job training (OJT) for IV&V personnel under IV&V masters were discussed. In some cases, some organizations had very strong negative side effects. These were also addressed. Some of these included poor morale, tools that are not known or understood, and the wrong use of a tool in specific situations.

The final issue addressed the Independence of the IV&V group from the PM. In all cases the IV&V group was assumed independent of the software developer. It was the general consensus of the subpanel that independence from the PM was a positive factor. For the most part PM's are concerned with several factors that may blind them in analyzing the effects of problems discovered by the IV&V group. Noting that there are several perspectives, the capability of the IV&V group to independently raise problems to higher levels of management was considered a significant benefit. This situation would only occur if the IV&V group is independent of the PM in the chain of command.

As mentioned earlier, there are several interdependencies among these factors. The financial resources available effect the capability of the IV&V activity. Critical DOD programs require additional IV&V and thus require more financial resources and highly skilled IV&V personnel. Highly critical programs have a need for the IV&V agent to discuss problems with management above the PM if the IV&V group's recommendations are ignored due to other pressing issues in the eyes of the PM. This analysis is not meant to be exhaustive. It was written to provide a better understanding of the analysis that the IV&V subpanel on IV&V in the organization accomplished during the workshop.

ALTERNATE MANAGEMENT MODELS FOR IV&V

Numerous organizational structures exist for the accomplishment of IV&V. To address all such structures in detail would be impossible. However, most of those structures may be described generically from the standpoint of a limited number of management models. Two management models, relating to IV&V efforts, are briefly addressed here in terms of the basic issues described above and the associated control methods such as command, operational, and funding control. Most other management models are variants on the two models discussed.

The intent of this section is to offer the program manager examples of specific management models and associated advantages or disadvantages in terms of the basic issues discussed above. He, then, may extrapolate from these models a method to attack his specific situation and, ultimately, facilitate his decisions to allocate IV&V tasks to particular resources available to him.

PROGRAM MANAGER MODEL

Model 1 (Fig 4.2-2)

This model is based on the assumption that the IV&V effort is totally under the control of the Program Manager (PM). The PM will decide the levels of IV&V to be performed, the agency (or agencies) to perform the IV&V, and how the results are used. Various options exist for selection of IV&V personnel (direct staff, Matrix, Test Activity or Contractor) but all share the common trait that the personnel are directly responsible to the PM and have no external reporting vehicle.

This model is basically the "Status Quo" in many organizations. The IV&V activities are thus dependent on the PM's view of its importance and budget considerations. While policy and training could raise the level of PM's awareness of the need for IV&V, it would still have to fit within the overall budget, and be subject to PM action to operate on the IV&V results. Independence is only measured from the developer/contractor and not from the developing agency.

CAPABILITY

IV&V functional capabilities and personnel training and skill levels are an integral part of this issue. Based on this model, the ability to respond to this issue area was ranked low to medium. Rationale for this decision centers on the assumptions that the IV&V group is temporary, may lack experience in the test arena, is task driven by the PM and may not have access to adequate software support tools.

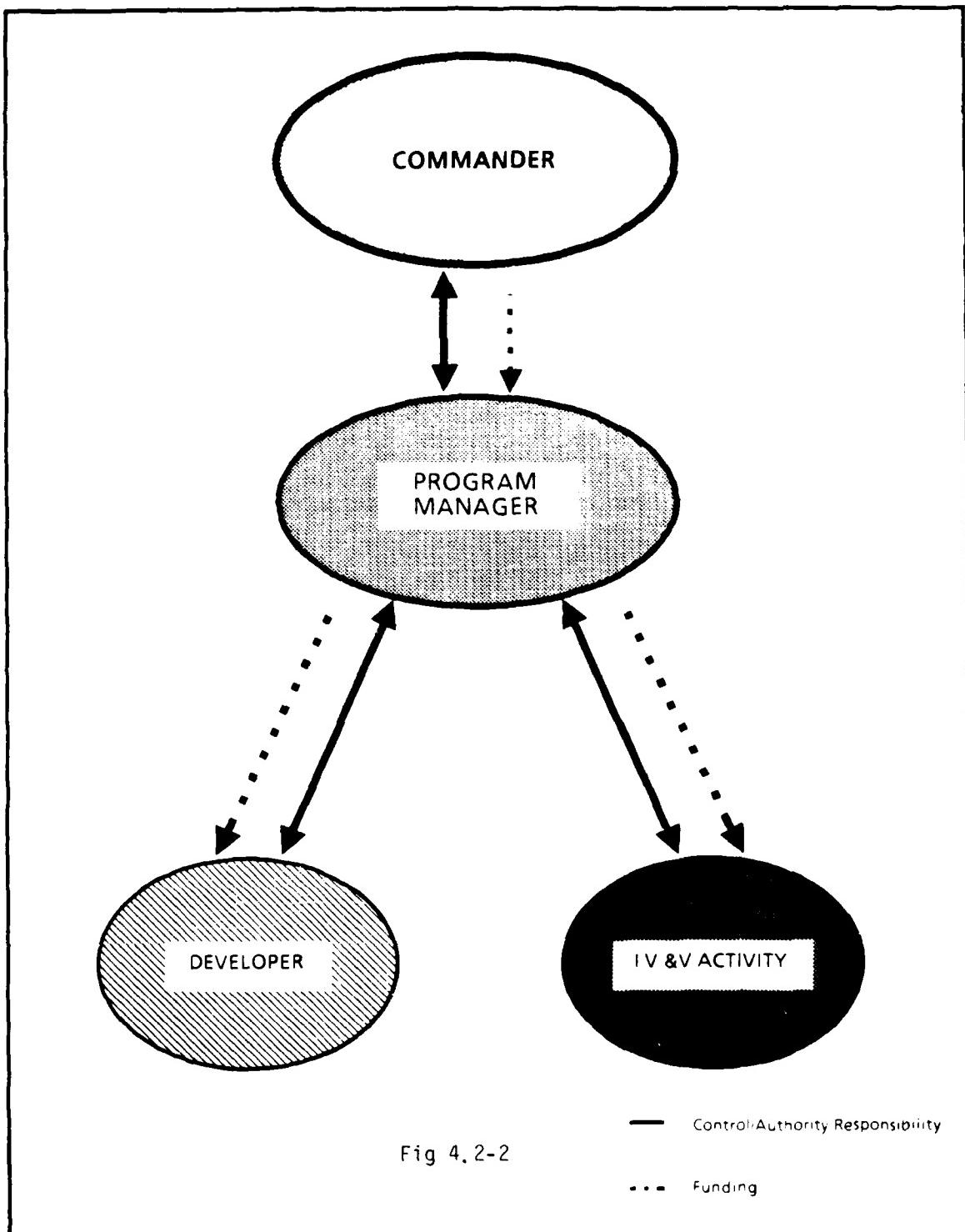
Personnel training may cover a period of 2 to 3 years depending upon the embedded computer system (ECS) complexity. The PM must therefore plan far enough in advance to staff the IV&V group enabling the group to be functionally capable to perform IV&V tasks.

RESOURCES

Resources include funds, personnel, facilities, facility location, tools and security. Since scope, objectives and goals of the IV&V effort are set by the PM, the span of resource development could range from maximum to minimum depending upon the PM V&V philosophy. This issue was therefore ranked medium.

Funds are the primary resource with personnel, facilities, facility location, tools and security being secondary and dependent upon funds. Generally, when program funds are short, the IV&V level of effort will be reduced.

The PM must actively seek personnel to be matrixed into the Program Management Office (PMO) to support each IV&V effort. This problem is two fold. The PM may have difficulty in funding qualified individuals and then, if found, qualified personnel may be reluctant to work under an inflexible model where the PM sets policy and procedure.



Program Manager Model 1

Facilities, facility location and tools can be grouped together. Under this management model, the PM is solely responsible for establishing facility location, specifying facility requirements and then defining the software support tools to perform the IV&V effort.

The PM must also specify security requirements. If a secure environment is required, the PM must budget for and develop a secure facility.

TIME

Response time and schedules are grouped into this issue. Time has been viewed from two points in this model. When viewing time as a PM one conclusion is reached which is totally different from time as viewed by the IV&V group.

When considering time as a PM, it is ranked medium to high. That is, the PM has a high degree of control over response times and schedules. As an example, software delivery dates are normally not slipped as a result of not completing the test phase. Instead, the scope of testing is varied to meet the shipping date.

This in turn causes the time issue to be ranked low by the IV&V group. This group has no control over response time and schedules. Their hands are essentially tied by the PM.

CRITICALITY

This had little differential impact between the two models. The only difference would be the view of the criticality of the software (and its IV&V) to the overall project. A PM without background in software acquisition may not understand the importance of IV&V to the process. In this case the PM model would have a low rating.

SIDE EFFECTS

This model results in a medium level of side benefits to the Government. Depending on implementation, learning curve benefits may not be available to either the next project or to the PDSS. The need to "Re-invent the wheel" during the IV&V process is a real possibility. Transfer of IV&V tools and corporate knowledge to the PDSS agency is not facilitated.

On the other hand, with the IV&V effort totally under the control of the PM, integration of the IV&V effort into the overall plan can reduce asset utilization and its cost. Duplicative testing may be reduced. This model provides potential short term side benefits at the expense of the longer term.

INDEPENDENCE

This model has a low level of independence from an overall Government standpoint. In the best case, the IV&V personnel would be given adequate resources and importance to accomplish the task. Financial and programmatic considerations could force a reduction in the effort or repression of the results. Without any independent appeal to a high authority, the goals of an IV&V effort are easily lost.

IV&V ENTITY MODEL

Model 2 (Fig 4.2-3)

The PM/PDSS organization receives guidance, responsibility, and authority from a higher manager or commander. The PM/PDSS organization also receives adequate funds to perform his function. The PM/PDSS organization provides guidance, responsibility, authority, and adequate funding to a doing organization. In most cases during development the doing organization is a contractor, but could also be internal.

The IV&V entity. At the same level of command as the PM/PDSS organization is a group identified as the IV&V entity. This formal, in-house entity is responsible for performing IV&V on multiple projects. IV&V is performed by this in-house entity or an appropriate mix of government/industry workforce responsible to the in-house IV&V entity.

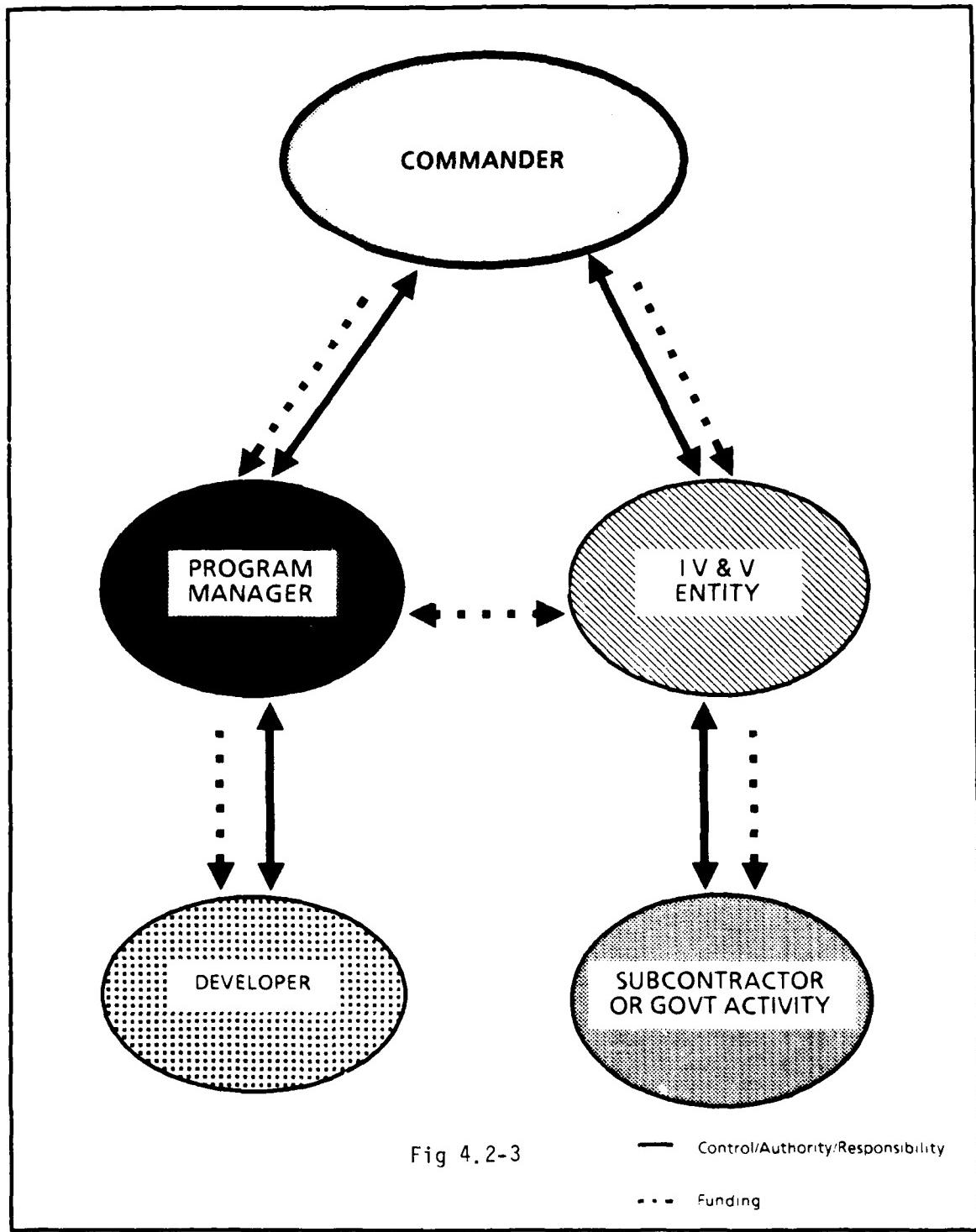
The IV&V entity, because of its formal structure has a staff function which includes generating IV&V policy, research and development of tools, techniques, facilities, and training to advance the state-of-the-art of IV&V. Funding for the IV&V entity is twofold. The bulk of the funds are provided by the individual PM/PDSS organizations to support their individual projects. As such, guidance, responsibility and authority are provided with this funding. Independent assessments of the doing organization are provided to the PM. The IV&V entity also receives guidance, responsibility, authority and the remaining portion of its funding from the higher manager or commander.

Assessments, independent of the PM are provided directly to the higher manager or commander. This feedback loop is not expected to be used routinely but must be there to assure that the PM is giving adequate management attention to the IV&V entity. Otherwise the PM could bury findings/problems until later in the life cycle when they would be more costly to remedy.

The IV&V sub-contractor/government entity. The IV&V contractor(s)/government entity(ies) are the appropriate mix of government and industry workforce to support all work efforts the IV&V entity is responsible for performing. These entities have specific statements of work similar to mission/functions of in-house entities. These entities are transparent to the PM. Guidance, authority and funding are received from the IV&V entity.

CAPABILITIES

In terms of capability, this management model is generally more powerful than other models in that the IV&V entity can usually attract and train talented individuals who have or will obtain IV&V experience. Secondly, variations of workload and complexity of projects can be more readily accommodated because the work effort can be distributed within the IV&V entity or contracted to companies which can handle the workload complexity problems.



IV&V Entity Model

The functions performed by the IV&V entity can be tailored to a project within the overall constraints (capabilities, tools, approach, methodologies) of the IV&V entity. This tailoring makes the IV&V activities specific for each project and consistent for multiple projects within a command or service. In addition, this consistency is supported by the corporate memory provided by the IV&V entity. This corporate memory includes the task, facilities and potential for training across multiple projects.

On the other hand, tailoring of the capabilities or functions of the IV&V entity which make up its corporate memory may not be as efficient for a specific project as having a totally dedicated, program manager group performing IV&V.

RESOURCES

In this model the IV&V support to the Program Manager (PM) is under funding control of the PM. The amount of effort for a specific project can be tailored by the PM who in turn, negotiates with the IV&V entity for the amount of effort and cost of the effort commensurate with the project goals and objectives. There is a possibility that the IV&V entity, which enjoys a second funding source, could supplement the PM if project funding is inadequate to perform requisite IV&V functions.

As with capabilities, resources benefit from the ability to accommodate varying workloads.

The additional funding source allows the possibility of research and development within the IV&V entity to enhance the state-of-the-art of IV&V techniques, tools and facilities while permitting exploitation of internal R&D efforts of contractors without regard to data rights. Here also, is the potential for competitively selecting only those contractors who are leading the state-of-the-art.

Disadvantages for this issue include the possible loss of control of the PM if the IV&V entity forces higher management intervention to obtain IV&V objectives. There is also a chance that the separate funding for the IV&V entity is derived from a tax or burden on multiple PM's, thereby taxing the PM's twice. It is recommended that a funding line separate from the PM's be sought to provide partial independence for the IV&V entity.

Security, location, and data rights do not appear a problem within the model.

TIME

This management model has the capability to be more responsive to the PM with regard to turnaround time and reducing the learning curve. This is because the formally established IV&V entity has existing skills, techniques, tools, etc., in place that could be readily applied to a specific project. On the other hand, a specific PM may not receive as quick a response as he would like because the skills, techniques, tools, etc. must be balanced against the requests of many PMs.

CRITICALITY

Variations of workload and complexity of projects can be more readily accommodated because the work effort can be distributed within the IV&V entity or contracted to companies which can handle the workload complexity problems. This allows appropriate work effort to be assigned to individuals or groups with the talent to most efficiently perform the work.

SIDE EFFECTS

There is a good opportunity, because an IV&V entity remains in existence for multiple projects, to offer training to organic DOD personnel by exposing those personnel to contractor techniques and tools. This training may be extended to PDSS groups if they are involved with the IV&V activities during development.

INDEPENDENCE

Independence is essentially two fold in this management model. At the lowest level, the IV&V entity performs and provides independent assessments of the doing organization and provides it to the PM. At the highest level, the IV&V entity performs an independent assessment of the PM which is provided to the higher manager or commander. This feedback loop is not expected to be used routinely but exists so appropriate management attention is provided to IV&V findings and problems. The obvious disadvantage of this model is that the PM loses some control of his project because of the IV&V link to the commander. This may force the PM to use a standard amount of IV&V. The other disadvantage is that it is difficult for the IV&V entity to divide its assessment responsibilities, and can lead to antagonism between the IV&V entity and the PM.

In terms of advantages and disadvantages, model number 2 appears to be superior to the first model from the perspective of the IV&V entity, program manager, and service. Most of the advantages to this model are retained when the project transitions to PDSS environment and the PM position is replaced by the PDSS manager.

CONCLUSION

The "IV&V in the organization" questions can be grouped into two broad categories:

- o Management concepts for actually doing IV&V efforts
- o How these concepts affect the relationships between the various entities involved in a software development.

Any organization with the requisite level of competence can conduct IV&V provided it is not part of the software development, organizationally or contractually. There is no reason that IV&V efforts cannot be done "in-house", in fact, the primary positive attributes for doing IV&V is experience in doing IV&Vs and access to IV&V tools and techniques, neither of which is confined to "in-house" or external organizations.

In general, PDSS personnel should be involved in both the software development and accompanying IV&V efforts. This involvement must be at some minimum level of effort and continuity to be worthwhile. The details of these involvements must be determined jointly among the PM, the PDSS and the primary IV&V manager. PDSS activity should be allowed to place the same amount of emphasis on IV&V involvements as on ongoing PDSS efforts.

The necessity to provide for IV&V of the PDSS efforts is primarily dependent on the scope of the "support effort" being undertaken. In general, PDSS efforts can provide a portion of the V&V activity from within the PDSS organization, or via some combination of the PDSS organization and the user. User involvement in V&V requires that individual support effort be carefully scoped prior to its initiation.

The roles, relationships and responsibilities of the various entities involved in a software development are as varied and complex as the developments themselves. It is easy to become buried in organizational details and/or peculiarities that may be successful or unsuccessful depending on the situation or individual perception. There do appear, however, to be two basic conceptual "management models" that can be reasonably addressed as concerns IV&V. These two models differ by whether one considers IV&V to be essentially a program function, to be the responsibility of the program or project manager; or whether it is believed that software IV&V is an activity sufficiently specialized, and with sufficient "independence" requirements as to require an independent organizational place and an independent function.

The first concept is in concert with the idea of providing a PM all he needs to do a particular job, and holding him totally accountable. The second concept recognizes that IV&V can easily be regarded as a burden by a particular PM, and that a properly managed IV&V "organization" could theoretically attain IV&V expertise ("corporate memory") and tools that could reasonably be applied across program boundaries.

An analysis of the advantages and disadvantages of both concepts as applied across service boundaries, led to a group consensus that the separate IV&V organization entity offered the best hope for true independence and excellence in the IV&V arena.

RECOMMENDATION

1. That the JLC endorse the need for separate IV&V responsibilities within acquisition commands.
2. That the JLC recognize that to reduce duplication of expensive test facilities, the PDSS facility should be the preferred agent to conduct the IV&V for both development and PDSS phases.
4. That during PDSS, the PDSS agency must also set up a separate IV&V process to ensure independence of that process.

4.2.5 PANEL B CONCLUSIONS AND RECOMMENDATIONS

The following broad conclusions and recommendations have been derived from the subpanels' deliberations:

CONCLUSIONS

1. Independent Verification and Validation is beneficial based on a cost/benefit analysis. These benefits are quantifiable and should be considered in all programs.
2. It is beneficial to begin the IV&V effort as early as possible.
3. IV&V can and should be used in all phases of the software development life cycle. IV&V activities are the same in PDSS as in other phases of the life cycle. The level of activity should be determined using the same criteria in all phases.
4. The level of effort for IV&V can be measured on discrete levels based on specific criteria and levels of risk. Models can be developed which will give the PM specific guidance on how much IV&V to use.
5. IV&V must be adequately financed to support the level of effort decided upon.
6. IV&V can be done by a separate contractor or "in-house" as long as the IV&V agent is independent of the developer.
7. Experience in IV&V and possession of and experience with the proper tools is the best predictor of an organization's future success in an IV&V effort.
8. The PDSS activity should be involved in the IV&V effort as early in the development cycle as possible. The preferred agent to conduct IV&V is the PDSS activity.
9. The descriptions of IV&V activities by level identified by subpanel 2 in Appendix J should be merged with the material on the software development life cycle prepared by subpanel 3 and presented in Appendix K. Figure 4-2-4 illustrates the resulting relationship.

RECOMMENDATIONS

1. Modify the definition of IV&V in the JLC policies on Computer Resource Management and Software Quality Program in accordance with paragraph 1 of Appendix J:
2. JLC policy should state that PMs should determine the extent of IV&V effort to be used in their program as part of an overall program trade-off analysis. This policy should be incorporated as part of a DOD Directive or Instruction and made part of the acquisition process as a check off item for ARBs, DSARCs etc. Adopt the policy statement specified in Appendix I.

REQ & ANALYSIS (SYSTEM)	SFW & REQ ANALYSIS	PRELIM DESIGN	DETAILED DESIGN	CODE & UNIT TESTING	INTEG & TEST	SFW & TEST	SYST INTEG TEST	OT&S (SYSTEM) TEST
DEVELOPMENT ACTIVITIES								
SYST ENGR PLANNING COST ANAL ILS (CALCMPS) INTERFACES REQ TRACING SPEC PREP TEST PLANNING FSD CONTRACT SIMULATION PROTOTYPING STANDARD SEL SRSS/SDR OPS CONCEPT	COMPLETE SRS SSR UPDATE ICD UPDATE CRISP UPDATE CDPD UPDATE STDS UPDATE C.R. HDW SPECS UPDATE FCD IDENT TEST ENVIRONMENT DEFINITION AUTHENTICATION OF SRS	DESIGN (HDW & SPW) TEST PLANNING DATA BASE DESIGN MANUAL PREP FDL PDR TEST TOOL TEST ENVIRONMENT MENT DEFINITION AUTHENTICATION OF SRS	DESIGN (FINAL) TEST PLANNING DATA BASE DESIGN CDR	CODE COMPILE UNIT TEST TEST PLANNING INITIATE CM	INTEGRATE AND TEST PQTS TRR	FQT UPDATE DOC FCA	SUPPORT SYST TESTS PCA MODIFY AS REQUIRED	SUPPORT OT&S MODIFY AS REQUIRED
IV&V ACTIVITIES:								
BARE BONES:								
ANALYZE SYST REQUESTS	ANALYZE S/W REQUESTS	ANALYZE TEST PLANS	ANALYZE TEST PLANS					
ANALYZE S/W DEV PLANS								
PARTICIPATE IN REVIEWS:								
SSR	SSR/SDR	FDR	CDR		TRR	FQR		
LOW SET:								
BARE BONES PLUS								
ANALYZE S/W ARCH	ANALYZE S/W TOP LEVEL DESIGN	SPOT CHECK DETAILED DESIGN	SPOT CHECK CODE WALKTHROUGHS		SPOT CHECK TEST CONDUCT			
MODERATE SET:								
LOW SET PLUS								
ANALYZE SYST ARCH	ANALYZE S/W DETAILED DESIGN	ANALYZE S/W CRITICAL CODE	TEST CRITICAL FUNCTIONS		PARTICIPATE IN PCA	PARTICIPATE IN PCA		
MODERATE SET LESS TEST MONITORING PLUS								
ANALYZE/VERIFY ALL ALGORITHMS	ANALYZE ALL DEVELOPED CODE						TEST ALL S/W FUNCTIONS	
FULL BLOWN SET:								
TEST ALL S/W FUNCTIONS								
ANALYZE/VERIFY ALL ALGORITHMS	ANALYZE ALL DEVELOPED CODE						SIZE AND TIME SOFTWARE	

Figure 4.2-4: Relation Between IV&V Levels and the Software Life Cycle

3. A PM guidebook should be developed to help the program manager

- o Complete a cost/benefit analysis
- o Determine the level of IV&V to be done
- o Determine what IV&V efforts should be accomplished during various phases of the life cycle

4. JLC endorse further data collection for the cost/benefit model improvement and calibration activities to provide the PM with a more precise resource prediction capability.

5. JLC endorse further data collection for the refinement of the criterion model for selection of the levels of effort for IV&V. Further research is necessary in the areas of weighting schemes for levels of risk, weighting schemes for the criteria employed, developing a methodology for mapping the weighting results to levels of effort.

6. JLC endorse the need for separate IV&V responsibilities within the acquisition commands.

7. JLC endorse the need for further study as specified in each subpanel report and make these subjects of follow-on JLC workshops.

APPENDIX A
SUBPANEL PARTICIPANTS

1. The Effectiveness of IV&V

Steve Habblett - Chairman
Ed Records
Ron Emberton
Lynn Redington
Raymond Rubey
Jim Clark

2. Criteria for the Use of IV&V

Adam Shirvinski - Chairman
Marilyn Stewart
Manny Baker
Ronnie Martin
Jose Ramirez
Ralph San Antonio

3. IV&V and the Software Life Cycle

Mike Carlin - Chairman
Bob Berri
Allen Irwin
Frank Bartosik
Roger Scholten
Tom Conrad

4. IV&V in the Organization

James Sides - Chairman
Marshall Potter
Greg Stratton
Bob Sutphen
George Neeman
Matt Fisher

APPENDIX B
BIBLIOGRAPHY

1. Management Guide for Independent Verification and Validation (IV&V), Space Division (AFSC), August 1980
2. Joint Regulation, Managemet of Computer Resources in Defense Systems (Draft), Joint Logistics Commanders, 15 June 1983
3. Joint Policy, Software Quality Program (Draft), Joint Logistics Commanders, 1 October 1982
4. Softfair: A Conference on Software Development Tools, Techniques, and Alternatives, Proceedings of conference held 7/25-28/83, Arlington VA.
5. A Guidebook to Independent Verification and Validation, Logicon, Aug 81.
6. Software Validation, Verification, and Testing Technique and Tool Reference Guide, National Bureau of Standards Special Publication 500-93, Sep 82.
7. Planning for Software Validation, Verification, and Testing, National Bureau of Standards Special Publication 500-98, Nov 82.
8. The Role of a V&V Contractor in the Development of Data Systems, WP-SD-10-78-838, Teledyne Brown Engineering, Huntsville AL, Feb 79.
9. Reifer, D. J., Verification and Validation and Certification: A Software Acquisition Guidebook, TRW-SS-78-05, TRW, Redondo Beach CA, Sep 78.
10. Baker, E. R., Fisher, M. J., "A Software Quality Framework," Concepts, Vol. 5, No. 4, Autumn 82, pp 95-107.
11. A Software Engineering Environment for the Navy, Report of the NAVMAT SoftwareEngineering Environment Working Group, Naval Material Command, Washington, DC, 31 Mar 82.
12. Analysis of IV&V Data, RADC-TR-81-145, Jun 81.
13. Proceedings of the Sixth Annual Software Engineering Workshop, Goddard Space Flight Center, Greenbelt MD, 2 Dec 81. (In particular, paper by J. Page of CSC on "Evaluating the Effects of an Independent Verification and Validation Team").
14. Independent Verification and Validation of Computer Software: Methodology NASA/JPL, 9 Feb 83, JPL D-576

APPENDIX C
PANEL PRESENTATIONS

1. Presentation by Dr. E. R. Baker on MIL-STD-SQAM and the relation of IV&V to Software Quality Assessment and Measurement.

LOGICON

PRESENTATION TO THE PANEL
ON PDSS IV&V
ORLANDO I WORKSHOP
31 OCTOBER - 4 NOVEMBER 1983

E. R. BAKER, Ph.D.
MANAGER, PRODUCT ASSURANCE
STRATEGIC AND INFORMATION SYSTEMS DIVISION
LOGICON, INC.

MAJOR ISSUES

LOGICON

- SOFTWARE QUALITY PROGRAM CONCEPTS
- PDSS ENVIRONMENT
- IV&V/QA INTERFACE
- IV&V/SETA INTERFACE
- IV&V TOOLS TRANSITION

MAJOR ISSUES

LOGICON

- SOFTWARE QUALITY PROGRAM CONCEPTS
 - NEW DRAFT JLC STANDARDS APPLY TO PDSS
 - ORGANIZATION INDEPENDENT CONCEPT
 - IV&V SUBSET OF SQAM
 - CRITERIA FOR USING IV&V
 - CRITERIA FOR INDEPENDENCE

MAJOR ISSUES

LOGICON

- PDSS ENVIRONMENT
 - MIX OF FIXES, ENHANCEMENTS, NEW CAPABILITIES
 - NOT ALL PROJECTS NEED IV&V
 - RELATES TO
 - EXTENT OF IV&V
 - CRITERIA FOR INDEPENDENCE

MAJOR ISSUES

LOGICON

- IV&V/SQA INTERFACE
- IV&V IS A DISCIPLINE
- SQA IS AN ORGANIZATIONAL ENTITY
- NO NEED FOR CHECKERS TO CHECK THE CHECKERS
- SQAM SHOULD BE APPLIED TO IV&V TOOLS

MAJOR ISSUES

LOGICON

- IV&V TOOLS TRANSITION
 - DEVELOP AND QUALIFY TOOLS DURING SOFTWARE DEVELOPMENT
 - TRANSITION TOOLS WITH OPERATIONAL SOFTWARE
- QUALIFIED IV&V TOOLS AVAILABLE FOR PDSS EFFORT

MAJOR ISSUES

LOGICON

- IV&V/SEA INTERFACE
 - NEED FOR INDEPENDENCE
 - POTENTIAL FOR CONFLICT OF INTEREST

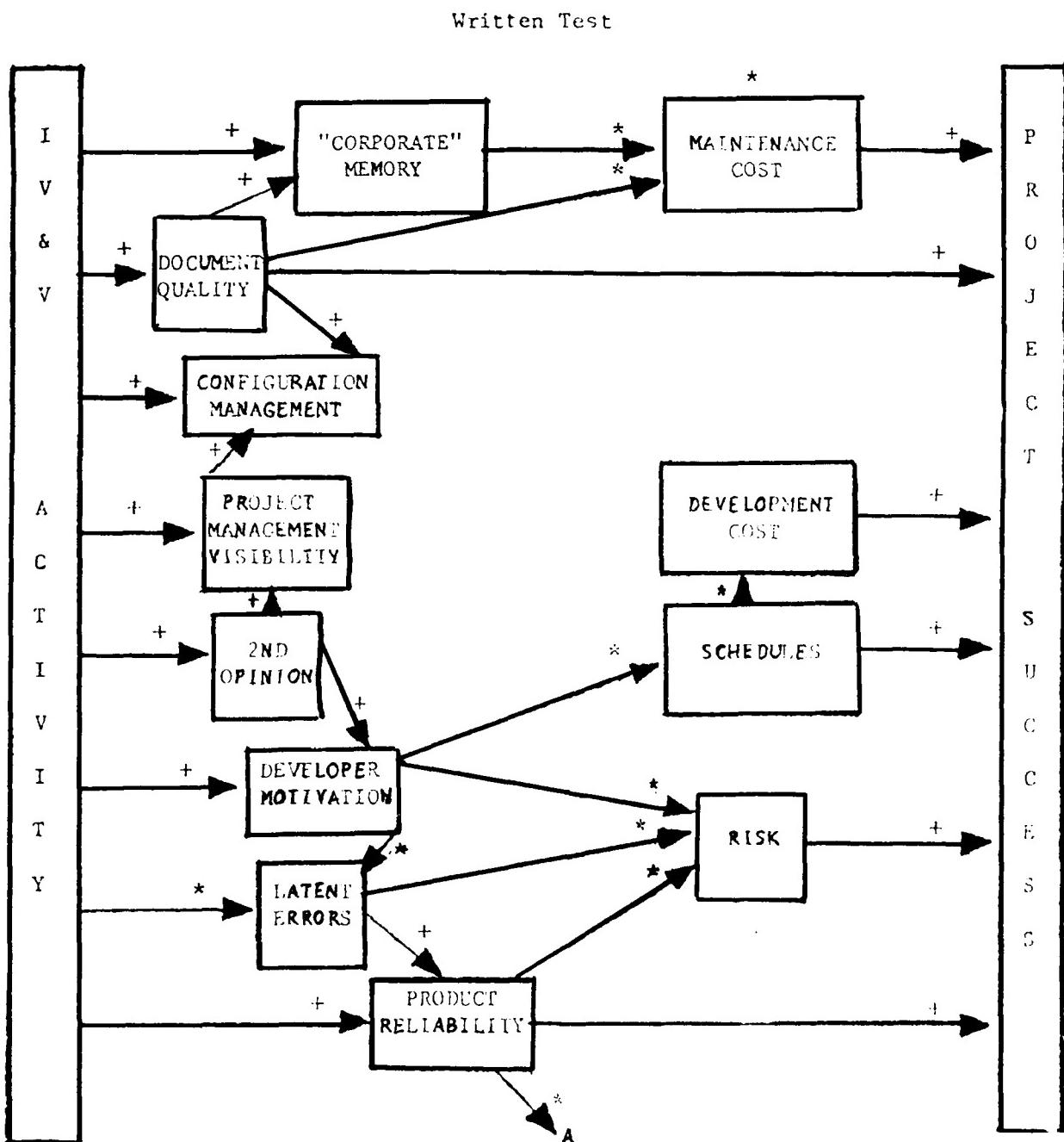
VIEW OF IV&V

LOGICON

- BENEFICIAL
- SUCCESSFUL PDSS IV&V EFFORTS
 - MINUTEMAN II
 - MINUTEMAN III
- ON-GOING
 - ARSIP
 - MM II OEP
 - MM II EPP
 - MM II MOTP

APPENDIX D
Benefit Interdependencies of IV&V

This diagram shows the interdependencies of potential benefits using IV&V from the inception of a project to its conclusion. The (+) signifies that an improvement can be realized or assistance can be gained. A (*) signifies that a cost reduction (\$ or time) can be realized or that a potential problem can be reduced.



APPENDIX E
IV&V Experience

Appendix E represents a subjective quantification of several IV&V projects and efforts which panel members have participated in. The data collected is based on specific projects, although, the identification of the project is not included in this report. The chart represents a matrix of IV&V "factors" related to the IV&V project. In addition, each "factor" is "referenced" to qualitative benefits listed in Attachment One. The net results of the panel's review of specific projects is that very positive qualitative benefits have generally been demonstrated. It was also concluded that quantitative methods for evaluating IV&V efforts on the sample projects did not exist.

The qualitative benefits listed in Attachment One are based on the subjective evaluation of members of the panel and are related to IV&V experience on specific projects.

ATTACHMENT ONE TO APPENDIX E

Benefits of IV&V

1. Warm and Fuzzy (confidence)
2. Earlier detection of errors
3. Latent errors become patent errors
4. Better documentation-requirements, design, code, test
5. Independent approach (2nd opinion)/technical evaluation
6. Government/prime/IV&V synergism
7. Corporate memory can be resident in IV&V team
8. Reduce risk
9. Evaluation of schedule and cost
10. Reduce schedule slippage - management insight into control process
11. Better structure for management reviews
12. Supports CM early audit (especially evolutionary development)
13. Improves management visibility
14. Helps the baseline freeze process
15. Reduces "black magic"
16. Motivation to prime
17. Helps decision makers make better decisions
18. Facilitates interoperability/integration

APPENDIX F
IV&V Case Study

The case study reflects an attempt to quantify the benefits of IV&V, with the end objective being a benefit to cost ratio. The system in the case study is an actual system which is near IOC and which represents actual quantified data collection. Very early in the project, software design problems created the need for IV&V and an IV&V contractor was hired. The IV&V contractor was able to influence software design philosophy and eliminated a forecasted six to nine month slip. Based upon a team of 50 software design engineers in place with the prime contractor at the height of the software design effort, a cost avoidance of \$1.8M was realized reflecting a benefit to cost ratio of 7.2. Thus, the government was able to save a minimum of \$1.8M by spending \$250K on an IV&V contractor. This case study is unique and certainly may not be applicable to all design efforts. However, it does represent one quantifiable example of very positive benefits for software IV&V. It also exhibits that program/project histories are a feasible approach for quantifying IV&V benefits.

4-2-F-1

ATTACHMENT ONE TO APPENDIX F

Results of a Case Study

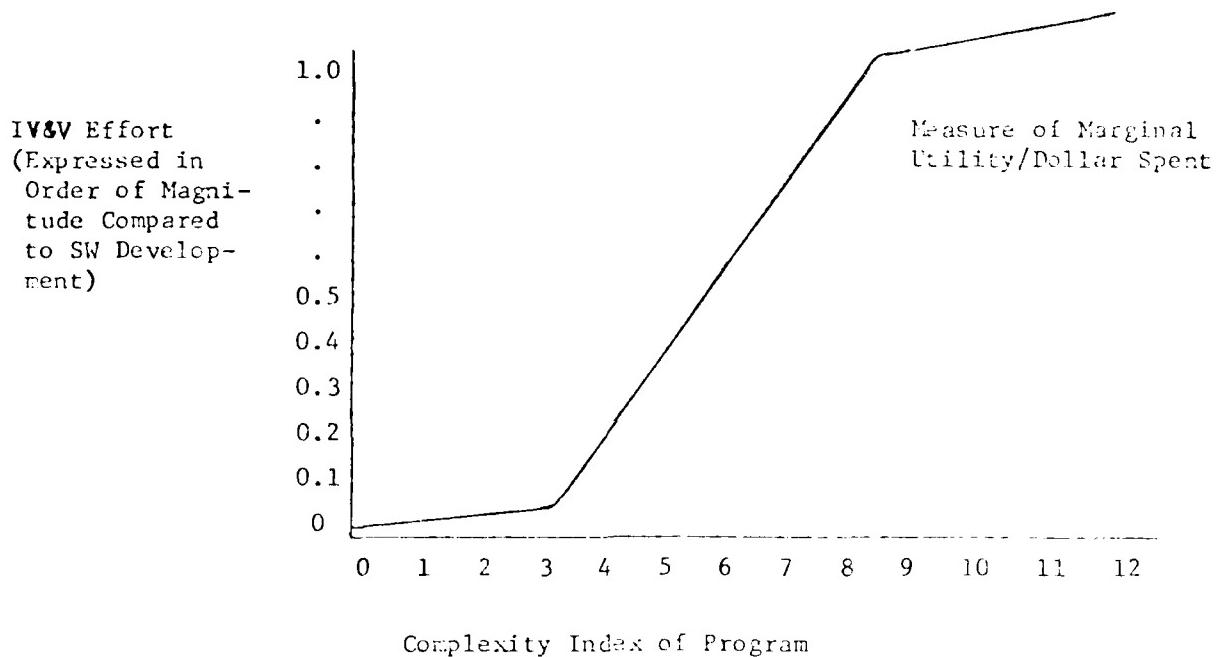
Automatic Satellite Utility System

- Planned Schedule 38 Months
- Predicted Cost \$6M
- Software Development
 - 20K lines
 - Front end problems
 - Forecasted 6-9 month program slip
- IV&V Contract
 - Began 6 months into the contract (32 mo)
 - Force new design philosophy on prime
 - Prepared computer resources management plan
 - Identified software development voids (manpower)
 - Identified specific design flaws
- IV&V Results
 - Prevented 6-9 month slip
 - IOC on time
- Cost Impact
 - Saved 25 manyears (\$1.8M)
 - Cost \$250K
- Benefit/Cost Ratio - $\$1.8M/\$250K = 7.2$

APPENDIX G

Model Descriptions

A number of models informally exist in industry which allow managers to estimate the amount of program resources to allocate to IV&V. The JLC should support further data collection, data reduction, model development and model calibration to provide a higher fidelity model for resource planning. A typical model is depicted below.



Complexity is a quantitative expression of software size, application, acquisition strategy risk, consequences of a SW failure, etc. Most programs fall between complexity indexes of 3 to 6.

APPENDIX H

Quantitative IV&V Cost/Benefit Analysis

An IV&V cost/benefit tradeoff analysis must consider both the quantitative and qualitative benefits. Quantitative benefits include both a reduction in the number of the software errors in the product during operational use and the earlier discovery of errors during the development phases. IV&V will be justified if the total cost of the IV&V effort is less than or equal to earlier error discovery plus the cost avoidance in operational use due to a more reliable software product. Stated in another form, the following relationship should prevail:

$$1. \quad C_y \leq (C_E) + (C_R)$$

Where:

C_y = Cost of IV&V effort

C_E = Cost avoidance in development due to early error discovery

C_R = Cost avoidance in operation due to software product reliability

An estimate of C_E can be obtained in the following manner:

N_{xy} = Number of errors made in phase x and found in phase y by the developer (where phase 1 is Requirements Definition, 2 is Design, 3 is Coding, 4 is Testing, and 5 is PDSS).

E_{xy} = Number of errors made in phase x by the developer and found in phase y by the IV&V organization.

C_{xy} = Average cost to correct an error made in phase x and found in phase y.

$$2. \quad C_t = \text{Cost without IV&V} = \sum_{x=1}^5 \sum_{y=1}^5 (N_{xy})(C_{xy})$$

$$3. \quad C_T = \text{Cost with IV&V} = \sum_{x=1}^5 \sum_{y=1}^5 (N_{xy} - E_{xy}) C_{xy}$$

Several existing studies indicate that a significant cost increase is associated with errors that are made early but discovered late in the development program. The following relationship is an expression of this cost growth:

$$4. \quad C_{xy} = (C_x)(10^{2p})$$

Where p = fraction of development time elapsed between when error was made and when it was discovered.

$$5. \quad C_E = C_t - C_T$$

In many cases, because of the cost growth factor expressed in equation 4 the early discovery of errors is by itself a sufficient justification for IV&V.

An estimate of C_R , the cost avoidance due to the increased software reliability achieved because of IV&V, can be obtained as follows:

6. $C_R = \text{Catastrophic error cost} + \text{serious error cost} + \text{moderate error cost.}$
7. Catastrophic error cost = $K_1 \times (\text{mission value} + \text{system value}) \times \text{probability of an error occurring} \times E_C$.
8. Serious error cost = $K_1 \times (\text{mission value}) \times \text{probability of an error occurring} \times E_S$.
9. Moderate error cost = $K_1 \times (\text{mission value}) \times \text{probability of an error occurring} \times E_m$

Where E_C , E_S , E_m are the number of latent catastrophic, serious, and moderate errors respectively. K_1 is the fraction of mission objectives achieved.

Some of the factors in the above expressions can be approximated as follows:

11. Probability of an error occurring = $\frac{\text{Number of missions planned}}{\text{Number of system functions, inputs and outputs}}$
12. Mission value = $\frac{\text{System Cost}}{\text{Number of missions planned}}$
13. $N_t = E_C + E_S + E_m$ (without IV&V)
14. $N_T = E_C + E_S + E_m$ (with IV&V)

Where N_t is the number of latent errors without IV&V and N_T is the number of latent errors with IV&V.

The effect of an IV&V effort is to reduce the number of latent errors (N_t is reduced to N_T). If this reduction is substantial and especially if the IV&V concentration is on catastrophic and serious errors, IV&V is justified for high-value systems and for "one-mission" systems (e.g., space boosters).

Each program can use the model, developing their inputs tailored to their system. A C_y factor less than or equal to the value of $C_E + C_R$ supports the use of IV&V. Sensitivity analysis should be performed when changes in assumed inputs will change the decision to use IV&V.

APPENDIX I

Proposed JLC Policy Statement Concerning IV&V

The PM shall, as a part of the PMD/PMP, determine if IV&V is to be applied to the software development activities. This determination shall be based on a cost/benefit analysis of the potential IV&V effort.

The potential IV&V effort shall be tailored to the software development environment and application.

For the purpose of this policy, the functions of the QA organization and the IV&V entity are not considered to be a duplication of effort.

APPENDIX J

IV&V Criteria - Supporting Material

1. Definition of IV&V

Independent Verification and Validation. The verification and validation of computer software performed by an organization that is managerially and financially independent from the developing organization.

Validation. The evaluation, integration, and test activities carried out at the system level to ensure that the finally developed CSCI satisfies the user's and supporter's requirements set down as performance and design criteria in the system and software requirements specifications.

Verification. The iterative process of determining whether the product of each step of the computer software development process fulfills all requirements levied by the previous step.

Note: IV&V Definitions have been modified from draft Joint Policy, Software Quality Program, dated 1 October 1982.

2. Goals of IV&V Levels

Bare Bones

1. Establish good requirements baseline.
2. Establish good development standards, procedures, and controls.
3. Perform thorough analysis of test program.

Low Set

1. Bare Bones, plus
2. Establish confidence in the development process.
3. Establish confidence in the Top Level Design.

Moderate Set

1. Low Set, plus
2. Establish confidence in Detailed Design.
3. Establish confidence in implementation of critical functions.

Full Blown

1. Establish high level of confidence in every aspect of the system.
(Note: A minority opinion held that IV&V addresses software only, hence full blown IV&V would "establish a high level of confidence in every aspect of the software.")

3. Criteria for Extent of Application of IV&V

The extent to which IV&V will be implemented on a program or project is based upon the extent to which the system (or the software, for a software project) is influenced by the criteria listed below. Note H stands for high risk, M for moderate risk, and L for low risk.

Criterion	Degree of Risk	Definition of Risks
a. Safety	H	Failure of software may cause catastrophic equipment damage or loss of life--includes: nuclear safety, range safety, flight safety of nonrated avionics, etc.
	M	Failure of software may contribute to equipment damage or personnel hazards--includes: controls and display indicators that may prompt incorrect commands, etc.
	L	Failure of software does not affect personnel or equipment.
b. Mission Essentiality	H	Potential error impact: mission failure.
	M	Potential error impact: degraded performance.
	L	Potential error impact: inconvenience.
c. Technical Complexity	H	Numerous interfaces, inputs, outputs, or system states; difficult-to-implement algorithms.
	M	A significant number of interfaces, inputs, outputs, or system states; moderately difficult to implement algorithms.
	L	An average or below average number of interfaces, inputs, outputs, or system states; algorithms not difficult to implement .
d. Type	H	Real-time, operational software.
	M	Non-real-time, operational software.
	L	Non-real-time, non-operational software.

e. Size	H	Over 100,000 lines of higher order language source code.
	M	Over 10,000 but less than 100,000 lines of higher order language source code.
	L	Less than 10,000 lines of higher order language source code.
f. Technology Required	H	Requires an advance in the state of the art or application of existing state-of-the-art to a new environment.
	M	Requires application of new requirements on an existing system.
	L	Transferring existing software to new hardware.
g. Degree of Generality	H	Very flexible: able to handle a broad range of inputs on different equipment; able to generate a broad range of outputs.
	M	Flexible input and output format.
	L	Restricted range of inputs or outputs.
h. Extent of Use	H	Defense Department, Worldwide, or Multi-Command.
	M	Single command or component command.
	L	Local or utility.
i. Supportability	H	No established support structure, considerable resources required for support, organic support.
	M	Support concept broadly defined but not specific to the system, moderate support resources required, organic support.
	L	Support concept specifically defined, stable.
j. Potential Cost/ Schedule Impact	H	Large program; complex, on critical path or may become critical path.
	M	Small program; complex or moderately complex, may or may not be on critical path.
	L	Off-the-shelf or noncomplex.

k. Security	H	Potential unauthorized access to classified data or unauthorized modification to CSCI or data base.
	M	Inadvertent loss or contamination of classified data base.
	L	No classified data involved.
l. Change in Requirements	H	Changes in requirements or objectives are continuous or frequent.
	M	Changes in requirements or objectives are occasional or infrequent.
	L	There are no changes in requirements or objectives.

4. Description of IV&V Levels

I. Bare Bones

A. Activities

- o Analyze system requirements, focusing on system requirements related to information processing.
- o Analyze software requirements, including interface requirements, for functional, performance, and qualitative adequacy.
- o Analyze the software development planning and procedures; spot check the developer's compliance with plans and procedures.
- o Analyze the software developer's software test plans and procedures to ensure that the developer's software testing is adequate.
- o Participation in technical interchange meetings and all Formal Reviews (SDR, SRR, SSR, PDR, CDR, TRR, FQR).
- o Develop IV&V plan

B. Typical Inputs

- o SDP, CM Plan, SQAM Plan
- o System/Segment Specification
- o Software Requirements Specification
- o Interface Requirements Specification
- o Software Test Plan
- o Software Test Descriptions

- o Software Test Procedures
- o Materials for Formal Reviews (SDR, SRR, SSR, PDR, CDR, TRR, FQR)
- o Access to Unit Development Folders

C. Typical Outputs

- o IV&V Plan
- o Technical Memoranda - evaluation of SDP, CM Plan, SQAM Plan
- o Traceability Matrices
 - System Requirements to Software Requirements
 - Software Requirements to Software Test Plans, Descriptions, and Procedures
- o Discrepancy Reports documenting anomalies in System Requirements, Software Requirements, Software Test Plans.
- o Technical Memoranda - evaluation of each Formal Review.
- o Final Report documenting IV&V findings and conclusions.

II. Low Set

A. Activities

- o All activities of "Bare Bones"
- o Analyze Software Top Level Design for completeness and adequacy in implementing Software Requirements.
- o Spot check detailed design and code walkthroughs
- o Spot check test conduct by witnessing selected software tests and analysing results.

B. Typical Inputs

- o All typical inputs of "Bare Bones".
- o Software Top Level Design Document.
- o Access to walkthroughs and test conduct.

C. Typical Outputs

- o All typical outputs of "Bare Bones".
- o Traceability Matrix - Software Requirements to Software Top Level Design.
- o Discrepancy Reports documenting anomalies in Software Top Level Design and software test.

III. Moderate Set

A. Activities

- o All activities of "Low Set".
- o Analyze Software Detailed Design for integrity and adequacy in implementing Software Top Level Design.
- o Extend System Requirements Verification to analyze system architecture.
- o Analyze code for integrity and compliance with design structures and standards; perform design reconstruction of critical code.
- o Perform independent tests of critical functions.
- o Verify accuracy and correctness of critical algorithms.
- o Participate in Formal Audits (FCA, PCA).

B. Typical Inputs

- o All typical inputs of "Low Set"
- o Software Detailed Design Document
- o Data Base Design Document
- o Interface Design Document
- o Preliminary code listings from Unit Development Folders
- o Object Programs for testing
- o Software Product Specification
- o Materials for Formal Audits (FCA, PCA)

C. Typical Outputs

- o All typical outputs of "Low Set".
- o Traceability Matrix - Software Top Level Design to Software Detailed Design, plus update to reflect Software Product Specification.
- o Discrepancy Reports documenting anomalies in Software Detailed Design, and Coding/Execution of critical functions.
- o IV&V Test Plan/Procedures for independent tests of critical functions.

- o IV&V Test Report for independent tests of critical functions
- o Technical Memoranda - evaluation of each Formal Audit.

IV. Full Blown

A. Activities

- o All activities of "Moderate Set" except IV&V organization typically does not monitor software development contractor's testing because IV&V contractor performs extensive independent testing.
- o Confirm capacity requirements for software by independent sizing and timing of software requirements.
- o Extend code analysis to include design reconstruction to all developed code.
- o Extend independent testing to include all software functions.
- o Extend algorithm analysis to include all algorithms; perform tradeoff studies and constraint analysis for critical algorithms.

B. Typical Inputs

- o All typical inputs of "Moderate Set".

C. Typical Outputs

- o All typical outputs of "Moderate Set".
- o Discrepancy Reports documenting anomalies in Coding/Execution of all software functions.
- o Expanded IV&V Test Plan/Procedures for independent tests of all software functions.
- o Expanded IV&V Test Report for independent tests of all software functions.

5. Strawman Correlation of IV&V Activities to IV&V Levels

The following table indicates activities performed in IV&V and their constituent subactivities. Also indicated is the IV&V level to which each subactivity is appropriate.

Key

- B = Bare Bones
- L = Low
- M = Moderate
- F = Full Blown

In certain cases, the IV&V level is indicated as "m/n", which means level "m" is appropriate for critical functions and level "n" is appropriate for all functions. In other cases, the IV&V level is indicated as "m, not n", which means that level "m" is the lowest level that is appropriate but at level "n" the subactivity is no longer appropriate.

<u>ACTIVITIES</u>	<u>IV&V LEVEL</u>
<u>System Requirements Verification</u>	
Consistency	B
Traceability	B
Interfaces (other systems)	B
Data Flow	B
Architecture	M
Quantitative Requirements	B
Testability	B
<u>Software Requirements Verification</u>	
Traceability	B
Functional Allocation	B
Control Flow	B
Data Flow	B
Sizing and Timing	F
Completeness	B
Consistency	B
Testability	B
Error Tolerance and Accuracy	B
Interfaces	B
<u>Top Level Design Verification</u>	
Structure	L
Interfaces	L
Traceability	L
Control Flow	L
Data Flow	L
Sizing and Timing	L
Completeness	L
Consistency	L
Error Tolerance	L
Global Data Definition	L
Accuracy	L
<u>Detailed Design Verification</u>	
Compliance with Standards and Conventions	M
Modularity	M
Interfaces	M
Traceability	M
PDL Analysis	M
Control Flow	M
Data Flow	M

<u>ACTIVITIES</u>	<u>IV&V LEVEL</u>
Sizing and Timing	M
Data Structure	M
Accuracy	M
Consistency	M
Completeness	M
Supportability	M
Walkthrough Evaluation (spot check)	L
<u>Code Verification</u>	
Standards Compliance	M
Design Compliance	M/F
Timing and Sizing	M
Accuracy	M
Traceability	M
Data Flow	M
Control Flow	M
Walkthrough Evaluation (spot check)	L
Unit Development Folders (spot check)	B
<u>Validation Testing</u>	
Independent Testing	M/F
Evaluate Integration Testing	B, not F
Evaluate Stress Testing	B, not F
Traceability	B
Monitoring (spot check by witnessing)	B, not F
Review plans, descriptions and procedures	B
Evaluate Performance Test	B, not F
<u>Algorithms Verification *</u>	
Derivation	M/F
Comparison with Standard Models	M/F
Tradeoff Studies	F/none
Accuracy Analysis	M/F
Constraint Analysis	F/none
* Note: If program office expertise/staffing is limited, these levels may be more appropriate to the "low" level, particularly if system engineering has not been extensive.	
<u>Documentation</u>	
Software Development Plan	B
CM Plan	B
SQAM Plan	S
ECPs, SPCRs	B

<u>ACTIVITIES</u>	<u>IV&V LEVEL</u>
<u>Review and Audit Participation</u>	
SDR	B
SRR	B
SSR	B
PDR	B
CDR	B
FCA	M
PCA	M
FQR	B
TRR	B
Technical Interchanges	B

APPENDIX K

LIFE Cycle Chart

1. Overview

Figure K-1 shows the various JLC approved activities of the software development life cycle to which the system development life cycle activities of system requirements analysis (A), system integration testing (H), and OT&E (I) have been appended. Each block is lettered. Those letters will be utilized in the following discussion and in Table K-A which shows the data required to be delivered by each activity block.

Figure K-1 is also divided into two parts, the upper part showing the activities performed by the developing agency or contractor for software, and the lower half showing the corresponding activities to be performed by the IV&V function. No attempt has been made to identify agency responsibility for IV&V or to what degree the IV&V is being performed (see subpanel 2 for degree of IV&V). The "evaluation" activity is more clearly defined by subpanel 2 and by the IV&V data delivery, Table K-A.

An assumption is made in Figure K-1 that the IV&V activity must acquire or develop new tools and that these tools must be produced in time to be utilized in the evaluation of the application software being developed. Therefore, the IV&V activities reflect also a compressed development cycle with the same activities as those required to develop the application software. There also may be a need for hardware tools, but Figure K-1 does not reflect that need. The asterisks show those repeated in a major change during the PDSS era (with no new tools required).

2. System Requirements Analysis (A)

Column A lists the typical activities of the development contractor during the demonstration and validation phase of the life cycle. These activities will culminate in a system design which will be approved as the allocated baseline for the system. The IV&V activities, e.g., evaluation, are further described in detail in the report of the subpanel report on "Criteria for the Use of IV&V". The IV&V agent will at this time determine the need for tools required to assist him in carrying out his evaluation activities. If such tools are not readily obtainable from a commercial source or perhaps from his own inventory, then he will necessarily have to specify and develop those tools.

3. Software Requirements Analysis (B)

The JLC software requirements analysis activity centers on completing the software requirement specification (SRS). In this process it is necessary to complete the related system engineering documentation and project plans that might be impacted by changes in the requirements stated in the final version of the SRS. The activity culminates in the software specification review (SSR). The IV&V activity is concerned with review of the SRS, the document updates, and the adequacy of the SSR. In addition, the requirements for new IV&V tools are finalized in specifications (SRS and related hardware B-specs).

4. Preliminary Design (C)

Column C lists those activities of the development contractor which lead to the preliminary design. In so doing, he will continue the evaluation of specifications from the software requirements specification to the software top level design document. During this process, he will identify the tools necessary to adequately test the design, and to define the environment in which testing will occur. These activities will culminate in the preliminary design review (PDR). The IV&V agent's evaluation activities during this period (see report from subpanel 2) will include the active design of tools identified in the previous activity columns, and the participation in the preliminary design review.

5. Detail Design (D)

During detail design, the preliminary design is extended and documented in the Software Detailed Design document. This document is sufficient to permit coding to start. Other documents (some optional) include manuals, unit development folders, data base design (if required), and the software test description. The design and the supporting documentation are reviewed at the critical design review (CDR). The IV&V activity evaluates these detail design products (see subpanel 2), and supports the CDR. In addition, tool construction and testing is accomplished (probably with some overlap to the development activities).

6. Code and Unit Testing (E)

The activities in column E will be conducted to translate the baselined specifications into code and to begin the testing process to ensure that the code actually performs as intended by the specifications and to initiate the development contractor's configuration management of the code being generated at the conclusion of this phase. The IV&V agent will engage in the evaluation activities (see report from subpanel 2) using the tools he has previously identified as being necessary to the process.

7. Integration and Test (F)

Each computer software configuration item (CSCI) within the system must be integrated and tested. Prior to starting this testing, an informal review is held to ascertain that the developing agency is prepared to conduct the activity. The testing is often formalized at the computer program component (CPC) level (when specified in the SRS) for critical modules or algorithms. For each of these, a preliminary qualification test (PQT) is conducted. The IV&V activity supports the review and monitors PQT testing. Independent testing also may be required. A general evaluation of the integration process (see subpanel 2) also is conducted. Formal delivery of IV&V tools can be effected in this area as required to support IV&V on the application software.

8. Software Performance Test (G)

These activities (column G) comprise the formal qualification testing (FQT) performed to demonstrate that the software system, as a whole, performs as intended by the specifications. Prior to starting this testing, a Test Readiness Review (TRR) is held to ascertain that the developing agency is prepared to conduct the activity. Results of this testing will ordinarily require that documentation be updated to reflect the consequent changes. The IV&V agent will, in addition to the evaluation activities listed in the report of subpanel 2, conduct some measure of testing independently to address particular aspects of the development which are deemed worthy of additional investigation or scrutiny, depending on the level of IV&V applied (see subpanel 2 and Appendix J)).

9. System Integration Test (H)

Subsequent to FQT of configuration items, these components are assembled into the system and the system-level tests are conducted by the development agency. Since software configuration items are developed in environments which do not fully reflect all system interfaces or conditions, it is possible to unveil errors in design or requirements during this period of testing. Prior to the start of system testing, a functional configuration audit (FCA) is conducted on the test results of the software performance test. The FCA results, plus problems encountered during system integration test, can establish a need to modify the CSCI. Subsequent to successful system integration test, each CSCI can be authenticated by the government through the physical configuration audit (PCA). The IV&V activity witnesses the system test, supports the PCA and tracks the successful incorporation of the required changes.

10. Operational Test and Evaluation (I)

In OT&E, the development contractor is principally interested in correcting any defects that, from the user's perspective, result in the system not being able to be used effectively in performing its intended mission. The principal role of the IV&V agent will be to evaluate the correction process to ensure that the corrected code performs as intended.

	<u>Development Data</u>	<u>IV&V Data</u>
<u>A</u>	Final System/Segment Spec B Specs (prelim) (including SRS) DSARC PLANS SDP CMP QAP (SQAMP) ICD/RTM System Test Plan Ops Concept ILSP (CRLCMP) LSA/RLA Plans Parts Control Reliability/Maintainability Plans Cost Estimate TEMP	<ul style="list-style-type: none"> o Reports on eval/assessments of A-spec (Partial) SRS SDP RTM (Partial) Test Plan (Software) Prototypes (e.g. algorithms) Ops Concept (Software) Reviews (SRRs/SDR) <p>o Report on tools on hand and new tools required.</p>
<u>B</u>	SRS (completed) Updates to <u>A</u> as required	<ul style="list-style-type: none"> o Reports (subset) as in <u>A</u> (add sizing and timing anal) o Hardware and SRS specs (tools)
<u>C</u>	Top Level Design Doc (Software) Test Plan (Software) Preliminary Manuals Update to <u>A</u> as required	<ul style="list-style-type: none"> o Reports (subset) as in <u>A</u> (add software manuals top level design doc software test plan) o Same as Dev <u>C</u> for tools
<u>D</u>	Sfw test descrip Sfw detail des doc Data base des doc Test procedures (ITDTP) Firmware manuals Manual updates or new UDFs Updates to <u>A</u> as required	<ul style="list-style-type: none"> o Reports (subset) as in <u>C</u> add new Dev <u>D</u> items o Same as Dev <u>D</u> for tools o Same as Dev <u>E</u> for tools o Same as Dev <u>F</u> for tools
<u>E</u>	Test results (reports) (optional) CM status Source/object code (for IV&V) Preliminary sfw test procedures Updates to <u>A</u> as required	<ul style="list-style-type: none"> o Reports on independent tests o Report on dev test doc o Reports on code analysis
<u>F</u>	Test reports (optional) FQT test procedures CM status	<ul style="list-style-type: none"> o Reports on dev-add <ul style="list-style-type: none"> - Test results - PQTS - TRR o PQT (independent) results

Table 4.2-K-A IV&V Data Delivery (Page 1)
K-4

	<u>Development Data</u>	<u>IV&V Data</u>
<u>G</u>	Product spec (SPS) Test reports VDD Final manuals	<ul style="list-style-type: none"> o Reports on dev add Dev <u>G</u> products o Dev <u>G</u> products for tools
<u>H</u>	Updates to all documents as required	<ul style="list-style-type: none"> o Reports on dev updates o Updates to tool documents as required
<u>I</u>	Same as <u>H</u>	Same as <u>H</u>

Table 4.2-K-A IV&V Data Delivery (Page 2)

Figure 4.2-K-1 IV&V and the Software Life Cycle

ORLANDO I

FINAL REPORT

PANEL C

COST OF OWNERSHIP

MARCH 12, 1984

CO-CHAIRMAN: Lt. Col James Riley
HQ AFSC/DLA
Andrews AFB
Washington, DC 20334
(301) 981-2482

CO-CHAIRMAN: Mr. G. (Gene) Sievert
Teledyne-Brown Engineering
300 Sparkman Dr.
Huntsville, AL 35807
(205) 532-1500

TABLE OF CONTENTS

Paragraph	Page
4.3.1 Introduction and Executive Summary	4-3-1
4.3.2 Background	4-3-2
4.3.3 Panel Approach	4-3-2
4.3.4 Significant Discussions and Issues	4-3-3
4.3.4.1 Discussions On The Validity of the EIA Cost Software Prediction	4-3-3
4.3.4.2 Discussions On The Use Of R&D Verses O&M Funding To Support PDSS Activities	4-3-6
4.3.5 Results and Conclusions	4-3-6
4.3.5.1 Estimated Total Costs Of Embedded Computer For DoD Systems By 1990	4-3-6
4.3.5.2 The Use of R&D and O&M Funds For PDSS Activities	4-3-7
4.3.5.3 Current Service Approaches To Post Deployment Software Support (PDSS)	4-3-7
4.3.5.4 A Strawman PDSS Charter	4-3-10
4.3.5.5 Facilities Required For Post Deployment Software	4-3-11
4.3.6 Recommendations	4-3-11
4.3.6.1 Recommendations Resulting From Panel Discussions	4-3-11
4.3.6.2 Recommendations Resulting From Sub-Panel Discussions	4-3-12
4.3.6.3 Cost-Saving Recommendations	4-3-13

APPENDICES

Appendix	Page
A. The EIA Study: DoD Digital Data Procession Study, A Ten-Year Forecast	4-3-A-i
B. Strawman PDSS Charter	4-3-B-i
C. Facilities Required for Post Deployment Software Support	4-3-C-i
D. List of Panel Members Arranged By Subpanel	4-3-D-i

4.3.1 INTRODUCTION AND EXECUTIVE SUMMARY

The Cost of Ownership panel was chartered with getting a handle on the true life cycle cost of ownership of DoD software; with identifying actions which can be taken under JLC auspices to make it possible to identify, track and control those costs; to investigate the utility and feasibility of a common DoD PDSS center charter and draft such a charter if appropriate; and to recommend to the JLC actions which, if taken by the services, might significantly reduce software ownership costs. The panel succeeded in meeting these goals. The approach initially described in the charter was closely followed, and resulted in several outstanding products and recommendations.

The point of departure for both panel and subpanel discussions was a series of four briefings on software ownership cost. Pat Mellin presented a briefing which was prepared in 1980 as a result of a study sponsored by the Electronics Industry Association (EIA) on the cost of DoD digital data processing. The conclusion of most interest to the panel was that the total annual cost of ownership of DoD embedded computer software would rise to approximately \$32 billion by 1990. This briefing was followed by three presentations on software costs within the services. The estimates based on Army and Navy data, presented by Gene Sievert and Bill Smith respectively, were arrived at by parametric analysis and were generally consistent with the EIA forecast. The Air Force presentation by Jerry Schmidt, on the other hand, reflected actual POM submissions based on projections of systems to be supported by both AFLC and the using commands (SAC, TAC, etc.). When these figures were adjusted for inflation and extrapolated to account for AFSC development costs, the Air Force number was significantly lower than the EIA projections would indicate.

This variance among estimates triggered a lively discussion which pervaded all further deliberations at the panel and subpanel levels and in fact spilled over into casual conversation. This intense concentration on the cost prediction issue ultimately enabled the panel to reach consensus in addressing the two panel-level goals:

- determine the credibility of DoD 1990 predicted embedded computer costs
- determine the cost of maintaining post-development embedded software systems.

The panel finally agreed unanimously that while the growth rate in embedded software in the short term will be as high as implied by the EIA study, that growth rate will not be sustained through the 1980's. Thus the \$32 billion estimate for 1990 is probably high. We also agreed that we do not currently have the data to offer an alternative figure to the \$32B, but that the PDSS portion of that cost would probably be between \$5B and \$7B in 1990. The panel wrote a recommendation to the JLC to sponsor activities to enable accurate tracking of future total life-cycle.

All major subpanel goals were achieved. One subpanel compared the current service approaches to many detailed PDSS activities, and concluded that despite some different views relative to management and funding procedures there is enough internal similarity to make a common PDSS center charter useful. A second subpanel drafted an excellent strawman for a common PDSS center charter. The third subpanel agreed upon and documented the physical facilities required by a generic

PDSS center, including requirements to address security considerations. Finally, a subpanel produced several outstanding recommendations for actions which can be taken during the system acquisition process to reduce the eventual overall cost of ownership.

4.3.2 BACKGROUND

The most common manifestation of "the software problem" is the rapidly rising cost of ownership, including both software acquisition cost and the expense of Post Development Software Support (PDSS). While estimates of future cost vary widely, the most publicized is that of the Electronic Industry Association, which predicted in 1980 that by 1990 the annual cost of ownership of software for DoD embedded systems could rise to \$32 billion. This would represent an order of magnitude increase over the decade. Moreover, some of those who participated in the EIA study have indicated that they believe that their estimate in 1980 was extremely conservative, and that the number may in fact exceed \$40 billion.

Many Army, Navy, and Air Force experts have taken exceptions to these estimates, but to date have offered little substantiated alternative data. The reality is that we will probably never know who was right, because there is very little chance that the EIA predicted level of funding will be made available for the purpose. The bottom line is that unless we can find ways to reduce software costs, we could lose some of the mission capabilities which drive us to digital systems.

Because of these divergent views, it is apparent that the EIA prediction must be analyzed and the underlying causes understood. Upon this basis, the accuracy of the prediction can be ascertained and ways to present an accurate update identified.

A secondary but related issue is the growing cost of maintaining developed software after it enters to DoD's inventory. The growing volume of software already developed is impacting the DoD's software budget now and if the EIA's predictions are correct, the cost of maintaining even more software will grow dramatically over the next few years.

At issue also is the role of the individual service's Post Development Software Support Centers (PDSSC). Software maintenance is traditionally assumed to be a combination of error correction and software enhancements starting after the software is developed. In terms of PDSSCs, this view causes unique problems. Traditionally in the United States, software maintenance is performed by a subset of the original group of programmers who developed it. In the case of the PDSSC however, a new group of programmers must maintain the software and frequently, they must do this function using inadequate documentation and without the benefit of the insights of the original developers.

Software enhancements performed in these centers may approach the level of effort of the original effort. This raises issues of the type of money used (R&D or O&M) and of the relationship with PDSSC should have with agencies which perform specialized functions (i.e., system testing, during the acquisition process).

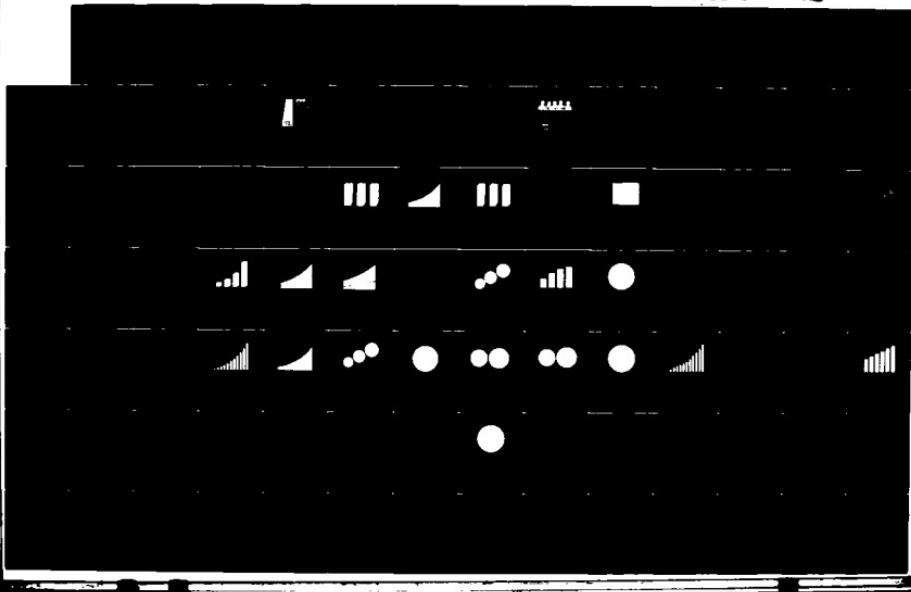
In summary, the major goal of the cost-of-ownership panel was to assess the credibility of the 1980 EIA estimate. Important secondary goals were to determine an appropriate charter for PDSS Centers and to use this charter as a basis for determining the estimated cost of PDSS Centers in terms of the relative levels

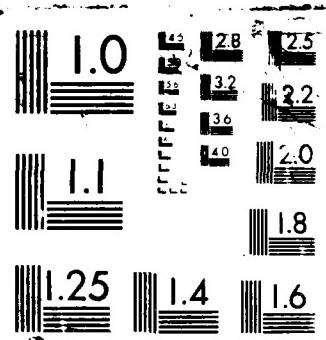
AD-8199 109 JOINT LOGISTICS COMMANDERS' WORKSHOP ON POST-DEPLOYMENT 3/6
SOFTWARE SUPPORT (U) JOINT POLICY COORDINATING GROUP
ON COMPUTER RESOURCE MANAGEMENT JUN 84

UNCLASSIFIED

F/G 12/3

NL





of O&M and R&D funding. A final goal was to identify innovative approaches which can potentially reduce these costs.

4.3.3 PANEL APPROACH

The approach used by this panel was to address the EIA cost prediction issue at the panel level and to utilize subpanels to achieve the secondary goals.

The EIA software cost projection was addressed using a two-step approach. The first step consisted of fact finding. This was accomplished by tasking individual panel members to research and estimate software costs through 1990 for each of the individual services. Another panel member was assigned to research the original EIA briefing. During the panel sessions, the original EIA briefing was repeated to the full panel, followed by briefings documenting the independent studies. The second step in the approach consisted of discussions of the briefings. Significant discussions are presented in Section 4.3.4 and the conclusion about the EIA cost estimate are documented in Section 4.3.5.1. The secondary goals of the panel were accomplished using subpanels. Each subpanel was assigned one of the secondary goals as a task and guided with a series of questions. The questions arranged by subpanel are as follows:

Subpanel 1: Current Service Approaches

How does each Service approach PDSS? What are the apparent advantages and disadvantages of each approach?

Subpanel 2: PDSS Charter

Is it possible to agree on a general PDSS charter? When in the acquisition life cycle should PDSS begin to play a role? Is there a PDSS life cycle which can be related to the product life cycle? What roles should PDSS play? Should these roles include system engineering and software development in support of ECPs?

Subpanel 3: Facilities Required

What PDSS facilities are required to support fielded systems? How realistically should PDSS facilities duplicate the real world?

What is the impact of security requirements (TEMPEST, software security, etc.,) on overall cost of ownership? Can technology help us?

Subpanel 4: Cost Saving Recommendations

Are there novel ways in which the costs of system acquisition or PDSS can be reduced?

Should funds be added during acquisition to increase the likelihood of competition for PDSS? Are such approaches cost effective?

Should PMOs (or SPOs) continue in existence after a system is fielded to play the role of the overall system manager or should PDSS take on this role?

4.3.4 SIGNIFICANT DISCUSSIONS & ISSUES

4.3.4.1 DISCUSSION ON THE VALIDITY OF THE EIA SOFTWARE COST PREDICTION

The discussions on the validity of the Electronics Industry Association (EIA) cost prediction occupied the panel throughout most of the week. The discussions had been preceded by a series of briefings on software ownership cost. Pat Mellin presented the EIA briefing which was originally prepared in 1980 and contained the conclusion that the total annual cost of ownership of DoD embedded computer software would rise to approximately 32 billion dollars by 1990. This briefing is presented in Appendix A. The briefing was followed by three presentations on software costs within the services. The estimates based on Army and Navy data, presented by Gene Sievert and Bill Smith respectively, were derived from parametric analysis and were generally consistent with the EIA forecast. The Air Force presentation by Jerry Schmidt, on the other hand, reflected actual POM submissions based on projections of systems to be supported by both AFLC and the using commands (SAC, TAC, etc.). The POM data was used as a basis for estimating the overall Air Force ECS budget based on DoD data that projects that software maintenance represents 70-75 percent of the entire software budget. The data was further adjusted to account for AFSC system developments scheduled for completion post 1990 and inflation (a factor of 1.9). This approach resulted in a forecast which was less than half of the EIA projections.

The variance among estimates triggered a lively discussion. Initially, the discussion focused on the Air Force briefing which was at variance with the conclusion of the other briefings. The thrust of these discussions revolved around two basic themes:

1. The completeness of the approach; the panel was afraid that major software development efforts had been missed.
2. The validity of the 30/70 rule (maintenance is 70 percent of the software budget).

In the case of the first theme, the panel did identify potentially missed software development but no where near enough to account for the discrepancy of the projection. In the case of the second theme, the panel concluded that there was no hard evidence to either substantiate or disprove the 30/70 rule.

Discussions on the other three briefings yielded the following points:

1. The three studies were all parametric based on assumed exponential growth rates.
2. The studies all included inflation as part of the growth. This effect increases precieved growth. It was noted that when normalized back to constant 1981 dollars, the EIA cost prediction for 1990 was closer to 17 billion dollars instead of 32 billion dollars.
3. The parameteric studies all neglect the realities of the congressional budget process and the availabilities of skilled people. These realities suggest that software growth can not continue exponentially. More realistically, the true curve is "S" shaped (see Figure 4.3-1) which will flatten into a growth rate approaching the growth rate

of the overall DoD budget. (It should be noted that the EIA study recognized these factors but did not directly factor them into the cost predictions.)

4. It was not clear in any of the briefings exactly what costs were included as "software" cost.

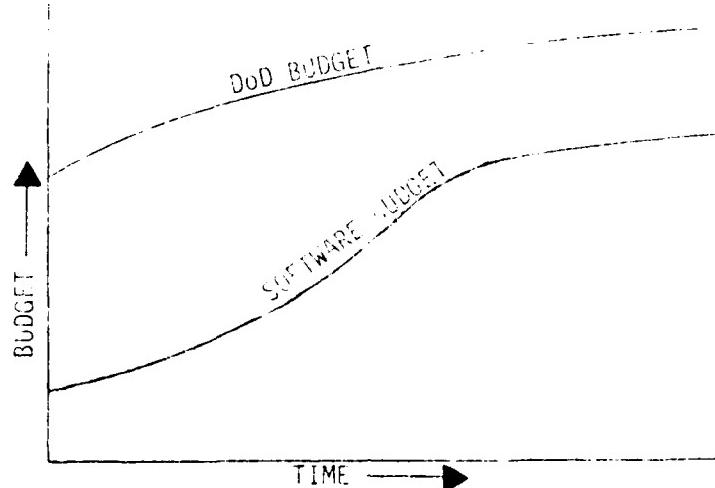


Figure 4.3-1 - Most Likely Growth Curve for Software Costs

Based on these discussions, the panel's attention began to focus directly on the EIA study. The key question was: What growth rate was assumed and what was its basis? The EIA study gave no indication of the answer. By normalizing the EIA software cost data in terms of constant 1981 dollars, it was observed that the average growth rate assumed for software costs was approximately 16 percent. It was hypothesized that the growth rate for the software budget was chosen on the basis of the hardware growth rate. Since software costs can be equated to the number of instructions produced, the two growth rates would in fact be nearly equal if software productively remained fairly constant and the average number of software instructions generated per computer also remained consistent with previous (before 1980) levels.

The key to the EIA prediction seemed to be the effect of the growth in the purchase of micro-computers in the DoD's budget. At the time of the EIA study, the growth in the purchase of micro-computers could have been accurately forecast. The panel reasoned that their use might not have been as evident. The following points came from the discussion:

1. Micros are being used as a substitute for engineering applications which were previously designed purely as hardware. That is, micros are being used to replace specialized hardware. Software is increasing but this may include a lot of "one time" software development. This growth is occurring because it is cost effective in the long run.

2. Micros are permitting a large number of new applications (i.e., digital displays, signal processing, etc.)
3. The number of software instructions developed per micro processor is considerable less than the number developed per larger machines. This is true even without considering the fact that the same software will be used many times over in the embedded computers during the production run of the system.

All of the previous discussions led the panel to the conclusions presented in Section 4.3.5.1 of this report. One of these conclusions, the conclusion that the EIA forecast was probably high, led to the final panel discussion on the cost of software ownership.

This discussion occurred when the panel was challenged to predict what the cost of software for embedded computers would be in 1990 in light of previous discussions. The panel discovered that it could not answer the challenge. In order to make a reasonable prediction against the hypothesized "S" shape curve, it would be necessary to accurately define DoD software costs in at least 4 years. The panel could not even begin to answer this challenge. When asked by the panel chairmen what the DoD had spent in fiscal year 1983, the panel concluded it did not know. As one panel member stated: "The DoD buys and predicts the costs of systems, not hardware or software. Software and hardware cost data is not kept but instead is rolled up into system cost data." The panels also admitted that contractors are not required to separate and report software costs and that the DoD did not know what it spent internally for software (i.e., DoD did not know how many of its own employees; hence their total salaries, travel, etc; were involved in contributing to the cost of software.) The panel concluded that the data did not exist for anyone to forecast the cost of software ownership. This discussion led to the recommendations presented in Section 4.3.6.1 of this report.

One final discussion was held to determine if the panel could predict the cost of providing post deployment software support in 1990. This discussion lasted less than one-half hour and the result is presented in Section 4.3.5.1 of this report.

4.3.4.2 DISCUSSION ON THE USE OF R&D VERSUS O&M FUNDING TO SUPPORT PDSS ACTIVITIES

The full panel conducted a brief but spirited discussion regarding problems within the services in determining proper funding appropriations for PDSS. Neither R&D nor O&M funding seems fully appropriate for PDSS function because PDSS is really evolutionary software development, but conducted after the formal system R&D had been completed. The services approach this dilemma in different ways, but it poses problems for all. The Navy and Air Force generally fund PDSS using O&M funds except in the case of a major system modification, in which case they often revert to a new R&D cycle. The Marine Corps used R&D funds throughout the life cycle. The Army uses a plethora of funding, ranging from numerous types of O&M, procurement, and R&D dollars. The problem appears most acute for the Army, but PDSS activities in all services frequently come under criticism for questionable use of appropriated funds for PDSS.

An additional problem is that of tracking PDSS funds. Using either R&D or O&M funding, funds are allocated for specific weapon systems, but are not

identifiable as PDSS funds unless specific procedural provisions are made within the weapon system program to do so. A third problem discussed (although no solution was offered) was the difficulty PDSS managers face in forecasting PDSS requirements seven years in the future in order to make realistic POM submittals.

4.3.5 RESULTS AND CONSLUSIONS

4.3.5.1 ESTIMATED TOTAL COSTS OF EMBEDDED COMPUTER SOFTWARE FOR DOD SYSTEMS BY 1990

The panel agreed that the EIA projection calcualted as approximately 16% annual real growth in software is probably high over the long run. While we did not have sufficient data to prepare a credible estimate of total 1990 software cost, the panel agreed that the PDSS portion of that cost should be in the \$5B - \$7B range in opportunity to reduce the total cost of systems by using software in their implementation. Thus, software will continue to grow as long as opportunities to reduce total systems cost are present.

In addition to implementation of previous hardware functions using software, several other factors will also drive accelerated software cost growth in the near term. Some of these are:

- Increased system complexity and flexibility
- Cheap memory resulting in larger programs
- Implementation of brand new functions (e.g. digital displays)
- Control of increasingly integrated functions and many others.

In contrast, there are at least two major factors which will tend to limit the software growth rate in the long term. One of these is the reality of the budget process, which, as the above factors become less dominant, will cause the rate of software growth to adjust to be more consistent with the growth rate of the DoD budget. There was some disagreement regarding the relative impact of this factor in consideration of the small proportion of the overall DoD budget visible as software funding. The other factor which should limit the rate of software cost growth is evolution in software development technology to increase productivity and quality.

Regardless of the true rate of software growth, even the most optimistic estimates represent a significant upward trend which requires measurement and management.

4.3.5.2 THE USE OF R&D AND O&M FUNDS FOR PDSS ACTIVITIES

There was unanimous agreement that a large part of the problem would be solved if we could receive common, officially sanctioned guidance to use either R&D or O&M funds for PDSS, if this guidance was clearly understood thoughout DoD and supported by the Congress.

It was also agreed that an even better solution would be to initiate a new category for appropriation devoted to the evolutionary development of software systems. This would solve the problem of tracking PDSS funding requirements for the

POM years by allowing aggregate funding of PDSS across weapon systems. The panel lost consensus, difficulty of justifying and establishing a new appropriation for evolutionary development might outweigh the advantages. These members were willing only to recommend that such an action be studied.

4.3.5.3 CURRENT SERVICE APPROACHES TO POST DEPLOYMENT SOFTWARE SUPPORT (PDSS)

To provide understanding, the following descriptions of PDSS Centers were used:

Army: The Army PDSS Center is a center within a DARCOM subordinate command established to support the software subsystems of all battlefield automated systems for which that command has logistics support responsibility. Each center normally supports numerous systems.

Navy: The Navy PDSS centers' functions and staffing are provided for as a subset of the In-Service Engineering Activity assigned life cycle support responsibility for the system. Note: that system may be an aircraft avionic package, a shipboard navigation system, or a shorebased C3I type system.

Air Force: The Air Force provides for a PDSS center as a part of Integration Support Facility (ISF) which is used to provide all hardware and software engineering support. This ISF is located in the engineering division or branch which supports the system program director (SPD).

Marine Corps: The Marine Corps has established a single PDSS center completely separate from hardware maintenance facilities. This center provides support for designated Marine Corps Software programs.

1. Organization Chain:

Within the services PDSS centers are located either in a logistics chain, a R&D chain or a combination of the two. The Navy has a combination chain with a single boss. The Air Force PDSS center is in the R&D chain, but receives direction from a logistics boss via the R&D chain. The Army established 11 PDSS centers located at the development commands, but funded by the readiness organizations within combined commands. Overall management of the PDSS effort is performed by DARCOM.

Coordination between R&D and logistics is always difficult. Having a single boss reduces the difficulty to some degree.

2. Development of Policy and Compliances:

Policy is developed at high level headquarters publish implementing instructions and ensure compliance by the PDSS centers within their individual commands.

3. How Funded:

The Air Force is O&M funded unless a major rebuild is required; then the system goes back to the developer and R&D funds are used. The Navy primarily uses O&M funds, but would also send major modifications back into a R&D cycle. The Marine Corps uses R&D funds. The Army uses a plethora of funding ranging from

numerous types of O&M, procurement and R&D dollars.

A standard approach to funding and a better definition of maintenance would help reduce some of these overly burdensome requisition and accounting functions.

4. Acquire Software Environment:

In all services, the PDSS centers, in conjunction with the developer, identify support requirements. In the Navy and Air Force the acquisition manager is responsible for procuring the initial suites of equipments, and the PDSS center is responsible for updating/replacing that equipment. In the Army, no defined responsibility exists which ensures that the developer acquires the support environment, including mockups and simulators.

5. How Location Is Determined:

The Air Force locates the PDSS centers within the system program directorate where sustaining engineering is also located. The Navy collocates the PDSS centers with the activity responsible for in-service engineering support. The Marine Corps only has one PDSS center whose command has the logistics responsibility for the system or has computer resources. In the event that the system is a command and control system, the PDSS center is collocated with the battlefield functional area school.

6. How System is Learned:

The Army and Air Force PDSS centers become involved at the beginning of the development cycle through either participation in the developmental process or by being the IV&V agency. The Navy may follow the same procedure, depending on when the PDSS center is designated. The Marine Corps PDSS center has previously been involved as part of the development responsibility and replace it with more of an IV&V type responsibility.

The involvement of the PDSS centers throughout the development cycle, commencing with Milestone I, is considered to be critical to the successful execution of performing PDSS work.

7. Use Of The PDSS Center For IV&V:

There is currently no stated requirement to perform IV&V in any of the services, and there is a wide variance of how the services accomplish IV&V. In those programs where there is a requirement for IV&V, the PDSS center is the most logical activity to do it and should be used to the maximum extent possible.

8. Software Configuration Control:

The PDSS centers of all services perform configuration control, but none are tasked with performing configuration management.

9. Type Of Changes:

There are basically three types of changes: those brought about by latent defects; those brought about by enhancement requests from the users; and major

product improvements. In all Services, the PDSS centers do the changes brought about by the first two. Changes brought about by the third are usually accomplished by a contractor, however, the PDSS center provides background and support. Currently, there is a need to establish common terminology to facilitate discussion and to aid in mutual problem solving.

10. Evaluation Of Complaint:

The Army maintenance directorate sends logistic support representatives to the user activity to investigate complaints. Once the problem is identified and verified to exist, the maintenance directorate notifies the PDSS centers who attempt to duplicate the problem. In the other Services, the PDSS centers receive the trouble report directly from the user and attempt to duplicate the problem. Responses back to the complaining user vary from periodic status reports about the complaint to not providing any follow-up information.

There should be some type of follow-up process in all Services.

11. Develop Software Engineering Change Solution:

In all Services, once the problem has been identified, the cause is explored by software engineers within the PDSS center. Solutions are developed and testing is conducted. This testing is to ensure that the original problem has in fact been solved and that additional problems have not been created.

12. Integration Testing:

In all Services, integration testing is performed when the PDSS center has completed system testing. With the exception of the Navy, testing is always performed upon the actual equipments which are being integrated. In the Navy, the size of the integration problem often prevents the PDSS center from conducting the integration testing in a totally real environment. The software which has been modified is run on actual system hardware, but those systems with which it communicates (i.e., integrated) may be simulated. The limitations upon integration testing of Navy shipboard combat systems due to the availability of actual systems are recognized and organic integration facilities have been or are being established. These facilities are outside the PDSS centers responsibility and control, but are available to the PDSS center for use.

13. Interoperability Testing:

All Services PDSS centers conduct interoperability testing to ensure that changes made to correct problems will in no way interfere with the capabilities of any systems to communicate with other systems.

14. Documentation Update:

All Service PDSS centers update documentation for every change made.

The major problem being experienced is the inadequacy of standards and resulting initial documentation. Standards must be published (SDS) that meet the needs of all services and contracts must be written to require documentation in accordance with these standards. Waivers must not be permitted.

15. Distribution Of Software Corrections To Users:

For systems where there is limited distribution, changes are hand-delivered and accompanied by sufficient instruction to allow the user to execute a smooth transition. In those cases where there is a large number of equipments in which changes must be installed, they are supplied through a distribution process along with a written instruction package.

4.3.5.4 A STRAWMAN PDSS CHARTER

A general PDSS charter is possible (See Appendix B) as a multiservice directive to establish roles, authority, responsibilities and lines of communication. The PDSS should begin playing a role in the acquisition life cycle as early as possible in the conceptual phase. To support this role, the PDSS must be designated officially as a PDSS not later than the decision point to go to full scale development. There is a PDSS life cycle which consists of phases, tasks and products that is similar to and supports the system product life cycle. The various roles of a PDSS should be specified by a PDSS charter which would be part of the official designation and consist of designation of PDSS manager, mission, authority and responsibilities, resource control, standardization for interoperability, communication channels, and location and support. The PDSS should include roles in systems engineering and software development to support analysis of discrepancies as to hardware vs. software, priority of the discrepancies (i.e., levels of mission/system criticality) also, to support membership on configuration control board (CCB) or engineering review board (ERB). Systems engineering and software development support to ECP's is a valid concept for those occasions when the PDSS has sufficient capability to support both maintenance functions and ECP development and the ERB/CCB has authorized the use of this capacity for ECP's.

4.3.5.5 FACILITIES REQUIRED FOR POST DEPLOYMENT SOFTWARE SUPPORT:

The facilities subpanel concluded that all service PDSS centers had common facility requirements. Their major conclusions were:

1. The PDSS Facility is an integral part of the mission critical system.
2. PDSS Centers should duplicate real world as close as possible.
3. Security requirement planning will increase PDSS facility costs in the short run but may reduce facility costs over its lifecycle.

A description of these facilities is presented in Appendix C of this report.

4.3.6 RECOMMENDATIONS TO THE JLC

4.3.6.1 RECOMMENDATIONS RESULTING FROM PANEL DISCUSSIONS

4.3.6.1.1 Software Life Cycle Cost Determination

The deliberations of the Cost of Ownership panel made one point embarrassingly obvious - that the full cost of ownership of embedded computer software in the DoD budget process makes these costs extremely difficult to

identify, especially during the R&D phase of the life cycle. Work breakdown structures adequate for attribution of development and support costs to software are rare. Contractor and government cost reporting systems are seldom structures to enable software cost determination.

The result is that software cost determination for any fiscal year and, consequently, accurate trend analysis, are impossible. The major impact of these inadequacies is that we lack the cost data base to confidently predict future software costs. To rectify this shortfall, the Cost of Ownership panel recommends that the JLC sponsor :

- a. A tri-service effort to identify the real cost of software for a near-term future baseline fiscal year.
- b. Changes in procurement regulations to force the use of work breakdown structures which clearly separate all software and system engineering tasks from hardware related tasks.
- c. Changes in contracting methodologies and procurement regulations to require contractors to report costs against these WBS'S.
- d. Changes in DoD accounting practices to make it possible to ascertain DoD software costs.

4.3.6.1.2 R&D vs. O&M Funding

As a result of the discussions of using R&D vs O&M funding for PDSS (see Sections 4.3.4.2 and 4.3.5.2), the panel makes the following recommendations to the JLC:

To solve the problem of multiple appropriations (R&D vs O&M) and funding lines to support software evolution after transition, a new funding line to provide for evolutionary support after transition should be establishing.

A minority of two panel members agreed that a new funding line for evolutionary software development would be the ideal solution, but felt that the difficulties in establishing a new appropriation could well outweigh the benefits. The recommendation of this minority was that the JLC sponsor a tradeoff study to balance the cost of justifying and establishing a new appropriation against its potential benefits.

4.3.6.1.3 Nature Of the "Software Explosion".

A recurring theme of the full panel discussions involved the nature of the predicted exponential increase in embedded computer software, and the extent to which this explosion should be viewed as an opportunity rather than as a threat. It was generally agreed that a large portion of the near term increase should be attributed to software implementation of functions which have formerly been implemented in hardware. The distinction is drawn against new functions which have only been made possible by advances in programmable digital systems. The point is that increasing software costs are not always detrimental, because conversion of functions from hardware to software implementations may well lower overall system

cost of ownership (or dramatically increase system effectiveness, flexibility, or readiness) while raising software costs. In these cases, increasing software cost might be desirable if the positive impact on the overall system is recognized. The panel recommended that the JLC highlight this positive aspect of software cost as follows:

DoD should direct its efforts to optimizing the expenditure of DoD resources even if it means rapidly expanding software growth rates.

4.3.6.2 RECOMMENDATIONS RESULTING FROM SUB-PANEL DISCUSSIONS

The following recommendation was proposed by the Charter Subpanel and approved by the entire panel:

The panel recognized the requirements for organic post/development software (PDSS) within all the services. There is a need DoD to promulgate standard functions and responsibilities of a post/ development software support activity (PDSSA) that can be fully implemented in each of the services. It is recommended by the panel that the JLC require the implementation, within all the services, for the PDSSA charter in Appendix B for the development, acquisition and support of embedded computer resources, including computer/processors, software and related hardware.

The following recommendation was proposed by the Facilities Subpanel and approved by the entire panel:

PDSS Facilities should incorporate existing or planned support components whenever cost effective over the lifecycle or when necessary for readiness.

4.3.6.3 Cost-Saving Recommendations

One sub-panel of the Cost of Ownership Panel was chartered to concentrate on ways to reduce software cost. This sub-panel produced the set of recommendations, approved by the entire panel which follow.

In general, the issues impacting software cost of ownership are much the same as those impacting software development cost, because the process of software maintenance is for the most part a process of continued development. Whatever improved the cost effectiveness of delivered software will usually improve the cost effectiveness of its maintenance.

Therefore, it is important that mechanisms which contribute to reduced cost of ownership be addressed as early as possible in the software life cycle and not later than contract requirements time.

Further, software cost reduction will never be a fully controllable and observable process unless all developments take place with full software cost accounting and measurement procedures in place and working.

There are many facets to software cost reduction which have been addressed under DoD STARS and other Service programs, and there is no advantage to discussing them all here. However, there are a number of issues relating to

software cost of ownership reduction that deserve attention because, from the perception of the Panel, they are of fundamental importance, and within the capacity of DoD and industry to tackle in the near term, and tend to be overlooked in the push to institute advanced technology programs in DoD.

ISSUE: Front End System/Software Acquisition Requirements

Most of software life cycle cost is determined by requirements laid down at development inception. Therefore, the start of a program must include all considerations for the life cycle support of that system. While sound acquisition principles governing system/software life cycle are expounded in theory within the material organization, they are frequently neglected in practice leaving the PDSS phase of the system holding the bag.

The request for proposal should identify all items needed for continued support throughout the total life cycle. Continued software support and evaluation requirements must be a key evaluation factor in making decisions regarding who is awarded given system contracts.

The cost/work breakdown structure must include the capability to track programs throughout the entire life cycle. All components must be planned for to be delivered as a part of the proposal -- either the support environment is furnished by the government or is deliverable under the contract. Likewise, all necessary test tools and development tools must be furnished or deliverable. In other words, a complete system must be obtainable from the contract which will allow full life cycle support and sustainment.

Further, the system hardware and software architecture must be configured to accommodate modular change or expansion over the full predicted life cycle of the system.

Investing front end resources in this way will definitely decrease the total life cycle cost.

Recommendation: JLC review policies governing acquisition requirements for adequate coverage of software life cycle support requirements and tighten procedures for promoting adherence to these policies.

ISSUE: Cost Data Collection, Cost Accounting And The Use Of Predictive Models For Software Costs

Much controversy is associated with the validity of the EIA prediction of \$34B to be spent on software support in 1990. In order to make any realistic projections, one must have more formalized and routine cost tracking and predicting mechanisms. The suggestion here is that costs be tracked by system (hardware and items) both with respect to efforts relating to software error correction, as well as software enhancements/modifications. These software efforts should be tracked by program element, e.g., OMA P2, R&D, or procurement appropriation account, whichever is applicable. A suggested format for this tracking system is given on the next page:

System/	AN/UYK-XXX	AN/UYK-XXY	TOTALS
Budget Category			
PROC APP	\$ or man-hours		
RDTE			
OMA P2			
OMA P7M			
TOTALS	Historical Cost Figures For FY XX		

Additionally, the current literature should be searched to determine how much work has been done to date and published in such journals as the IEEE Computer Magazine, the Journal of the ACM, NTIS, university theses, and other commonly available sources.

This issue is also directly related to the issue of proper funding policies within the Services. Additional work should be sponsored, either to be done in-house (by one of the Services or by OSD) or contractually, to develop/adapt analytical predictive models for estimating software costs as a function of system complexity, lines of code, life cycle phase of end item, etc. Models currently available include some based on the Raleigh-Norden distribution, various multivariate regression analysis models, the COCOMO model, etc. The use of these models should either be encouraged as predictive tools or new variants should be developed.

It is only through accurate data collection and extensive use of these analytical models will we ever get a thorough understanding of the future software support cost within the DoD.

ISSUE: Applications Software Reusability

The surest way to avoid software cost is to avoid developing new software. Reusability of existing software has the potential for significant software cost reduction (development and maintenance). In order to achieve maximum cost reduction benefits, a well disciplined approach to implementation must be followed.

Standards of module structure, as well as documentation (requirements, input/output, processing description) must be adhered to. Module function descriptions must be clear and concise so that libraries can be established to accomplish the clearinghouse functions needed to ensure maximum publicity. System architecture guidelines for improving reusable module insertion should be developed. Incentives should be provided for contractor/organic use of existing software modules and for the generation under reusability guidelines of modules not previously filed in the library.

This activity should dovetail with other ongoing standardization activities (e.g., Ada, documentation, ASPE, etc). A method of achieving this compatibility is by placing the definition/implementation/library functions under the software technology improvement efforts and assigning it to a centralized functional office for management.

The success of reusable software will be a function of how well the modules are documented, structured and how well the synopsis of module functions are publicized. An automated catalog (perhaps via ARPA net) must be considered. Definitions of incentives and requirements for use should be in the RFP. System architecture should be adapted for use with reusable software (where performance permits it).

Software reusability has been the subject of much discussion and study but little coordinated action has ensued.

Recommendation : JLC institutes a program to develop procedures, organization elements, policies and support tools necessary for reusability, and identify program areas of high software reusability potential to participate in such an initiative.

ISSUE: APSE Standardization

The EIA study predicting exponential increases in the cost of software has the attention of both DoD and industry. Whether one agrees or not with the dollar amounts, the consensus is that software cost trends must be changed and the costs brought under control.

As Dr. Martin has pointed out, one of the chief management requirements is effective control of costs and schedule.

One method of cost control has long been recognized. That method is standardization. The original intent of the development of Ada High Order Language was cost reduction through the control of HOL's and associated environments that would be reliable, adaptable, responsive, reusable and transportable. Unfortunately, it appears that DoD is about to lose control of its original intent -- cost control, through standardization -- as it attempts, to implement the Ada Environment.

Two of the three Services, the Army and Air Force have already embarked on full scale development of two different Ada environments: (1) Ada language system (ALS) by the Army, and (2) Ada Integrated Environment (AIE) by the Air Force. The Navy is adopting the Army ALS as the baseline for its environment.

Industry is rapidly gearing-up a proliferation of company-unique Ada environments.

Thus, the proliferation and non-standardization of Ada environments has already begun.

OSD and AJPO have recognized the potential problems associated with different environments and saw fit to establish interface teams with DoD and including industry and academia.

The situation and potential therefore exists for more fuel to be added to the exponential fire associated with escalating software costs rather than having a dampening effect.

The extensive development costs for multiple environments and the support costs to maintain those environments could well obviate most of, if not all, the anticipated cost savings and economic benefits envisioned or projected by the use of Ada. And while improved efficiencies may be generated via the STARS program, these efficiencies may well be lost in the additional expense of having to maintain and use separate non-standard environments.

Certainly the human element is the most costly aspect of software development and maintenance. DoD has justified STARS on the basis that if STARS is successful, the DoD will be able to decrease the human element somewhat and can reduce the projected exponential cost growth to one that is geometric. If separate Service teams must divert manpower and economic resources to the control and/or maintenance of multiple unique environments, not to mention having to, potentially, adapt to many incompatible industry Ada environments, the reduction of the cost growth scope from exponential to geometric is placed in jeopardy.

Recommendation : That OSD regain control and seriously examine the issue of Ada environments to determine the most cost effective method of continuing with the original intent of cost reduction through an Ada standard implementation.

ISSUE: Automation of Software Development Functions

The primary cost of software is people cost. There has been much attention given to trying to improve the efficiency of programmers in the software development process through training, structured code and other programming improvement mechanisms. However, there are limits to the degree of efficiency that can be achieved by individual programmers at developing each line of source HCL in a delivered system.

A direct and potentially much more effective approach to reduction of software cost is to eliminate, as much as possible, human involvement in the software production process through what are known as non-programming options such as reusable software, automated system generators, and very high level problem-oriented-language systems.

The Services should evaluate in-house and commercial tool developments that could be applied to automation of portions of the software development process from requirements through final testing. There are several areas which could benefit from this approach in the near term.

* Requirements specification/validation/tracking. A number of requirements statement languages and associated support tools of varying levels of maturity are available in industry. These systems help manage this highest cost leverage portion of the software life cycle.

* Applications generators/problem-oriented-languages. Such systems in well-defined application areas allow one or more stages of program specification, coding, and testing to be avoided along with the error attendant in the human element. Application generators go hand in hand with software reusability where predeveloped software building blocks can be used in the automated system generation

process.

* Documentation. Automated document generators can also generate test data sequences which check out system software components individually and in groups.

More ambitious solutions over the long term involve complete systems generation and testing directly from generalized "natural language" descriptions. While such approaches deserve continuing research, there appears to be little probability that they will impact life cycle cost prior to the 1990 time frame.

Recommendation : JLC identify specific program development areas which could benefit from application of available or near mature automation tools and begin to utilize these in specific applications hand in hand with cost data tracking and management.

ISSUE: Personnel Qualifications Should Meet (and not exceed) Minimum Acceptable Levels for the Appropriate Software Support

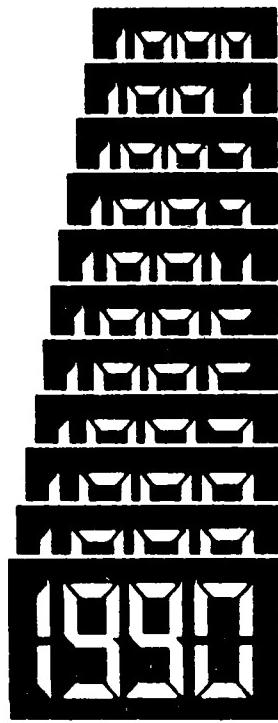
The thesis here is very straightforward. Commands should not use highly qualified system engineers to do routine coding and programming tasks. Likewise, they should not require these highly skilled personnel or their contractor support personnel for routine tasks.

This is not to diminish the importance of highly skilled systems engineering personnel, but only reinforce the commonly held management precept that one should accurately match the skills of the employees to the job or task at hand.

We have seen large discrepancies in the average "loaded" cost of software support personnel across PDSS centers and this is partly due to variations in the availability of certain skilled personnel in different parts of the country, but it is also due, in part, to requirements for personnel with skill levels higher than is absolutely necessary.

ISSUE: Coupling the Tech Base to Software User Requirements

There currently does not exist a formal mechanism for people in the support and sustainment of software to translate to the research and development community their needs for tools. A technique for tool development requirements needs to be started. We need a mechanism to allow these people to write a statement of need and for it to be evaluated and considered for research and development. A large payoff is available in the development of standard tools to be used in this area yet no mechanism exists for those "users" to pass on their needs in some kind of requirements document which becomes to some extent a formal requirement on the tech base community.



DOD DIGITAL DATA PROCESSING STUDY

**A
TEN-YEAR
FORECAST**

Appendix A

The EIA Study: DoD Digital Data
Processing Study, A Ten-Year Forecast

DOD DIGITAL DATA PROCESSING STUDY
A TEN YEAR FORECAST

Executive Summary

The "DOD Digital Data Processing Study - a Ten Year Forecast" was performed by an industry team under the auspices of the Requirements Committee, Government Division, Electronics Industries Association (EIA). The results of the year-long study was initially presented at the EIA Fall Symposium, "The DOD Electronics Market - Forecast for the 80's," which was held in Los Angeles on October 7-9, 1980.

The hypothesis behind the study was that an ever-increasing share of the DOD electronics budget is being earmarked for digital computers. The industry team, with representatives from Control Data Corporation, IBM, Intel, ROLM Corporation and TRW performed an analysis of the digital computer portion of the broader DOD electronics segment. The study included both Automated Data Processing (ADP) and the Embedded computer area; it included both hardware and software/services.

The study team used multiple sources to obtain and verify information including DOD budget data; congressional testimonies; over 40 personal interviews with experts in industry, DOD, congressional staff, OMB and GSA; periodicals; industry market research publications including Frost and Sullivan, DMS, Quantum, et al, and published data from several government sources including OMB, GSA and GAO.

The quantitative and qualitative results of the study are presented in this report. In summary form, a few of the highlights from this report are: (Unless otherwise stated all dollars are current in billions.)

- o Defense Electronics will increase from \$20.1 in FY80 to \$75.7 in FY90. Defense computers will increase from \$6.7 in FY80 to \$45.8 in FY90 - from 33 percent of Defense Electronics in FY80 to 60 percent by FY90.
- o Software and Services will increase from \$4.6 in FY80 to \$37.2 in FY90 - from 69 percent of the total Defense computer expenditures in FY80 to 81 percent by FY90.
- o Software hourly rates have nearly tripled since 1965 and are projected to be over five times the 1965 base by 1990. However, the cost of computer hardware is decreasing dramatically. By 1990, the cost of large mainframe computers and the cost of mini/micro computers are projected to be one-fifth and one-tenth respectively of the 1965 base.

- o In 1955, there were approximately 1000 computers and 10,000 programmers, a 1:10 ratio in the U.S. Today, there are approximately 900,000 computers and 240,000 programmers, a 9:24 ratio. Even with productivity improvements, the shortage of qualified software personnel will not end; software costs will continue to rapidly escalate.
- o During the 1980's:
 - The total DOD budget will increase 2.8 times,
 - The DOD Electronics budget will increase 3.8 times,
 - The DOD Computer budget will increase 6.8 times,
 - The DOD Software budget will increase 8.1 times.
- o ADP computers in Federal inventory will increase from 16,513 in FY80 to 58,070 in FY90. ADP computers in DOD inventory will increase from 6,435 in FY80 to 27,700 in FY90. During the 1980's, minicomputers will comprise a large portion of the Federal/DOD inventory. DOD's ADP hardware budget is forecast to increase from \$.8 in FY80 to \$2.7 in FY90; during the same period, the DOD software and services budget will increase from \$1.8 to \$5.2.
- o The ever-increasing DOD ADP budget combined with nearly constant in-house personnel levels results in an increasing percentage of DOD's ADP budget going to the private sector, as shown below:

FY	DOD ADP \$ To Private Sector	% Of Total ADP \$ To Private Sector
1978	\$ 926 M	48
1979	1,224 M	53
1980	1,482 M	57
1981	1,688 M	59

- o Embedded computers are defined in the study as specially designed, for example, designed to satisfy MIL-Specs, and are acquired as part of a total weapons package, thus "embedded" in a weapons system. It is not generally recognized by most personnel in the computer field that embedded computers presently represent over 60 percent of the DOD computer budget, and the percentage is projected to increase to approximately 75 percent by 1985 and 83 percent by 1990. Microprocessors will have an ever-increasing influence in the embedded area; much more so than in the ADP area.

- o Single chip microprocessors capable of performing a million instructions a second (1MIP) are forecast to be developed during the early 1980's.
- o It is forecast that in the coming decade, nearly every weapon system will have an embedded computer (or computers) somewhere in its control subsystem and/or C³I subsystem.
- o A larger portion of the embedded budget is returned to industry than from the ADP budget. An estimated 87 percent of the 1980 embedded budget was contracted to industry, most of which came from RDT&E accounts with smaller portion from O&M and procurement accounts. There is a definite trend for the services to function more and more as program managers executing contracts to industry in the embedded area as opposed to performing computer design/development tasks in-house.



THE STUDY TEAM

GD
CONTROL DATA
CORPORATION

DAVE STEPHAN

intel

DICK DAVIS

IBM

BILL BARBAZETTE

ROLM
CORPORATION

LINDA JOHNSON

TRW

BILL MURPHY

The study team consisted of Dave Stephan from Control Data Corporation, Dick Davis from Intel, Bill Barbazette from IBM, Linda Johnson from ROLM Corporation and Bill Murphy from TRW.

STUDY INCLUDES:

- **ADP & EMBEDDED**
 - **COMPUTER HARDWARE, SOFTWARE & SERVICES**
 - **FUNDING; RDT & E, PROCUREMENT, O & M**
-

NOT INCLUDED:

- **CLASSIFIED**
- **OFFICE EQUIPMENT eg WORD PROCESSING,
CALCULATORS, ETC.**
- **EXPENDABLE SMALL MUNITIONS**

The study includes: both Automated Data Processing and the Embedded computer area, computer hardware and the labor-intensive activities referred to as software and services. DOD funding sources identified in the study include RDT&E, Procurement and O&M. Emphasis has been given to the Army, Navy, and Air Force.

The scope of the study has been limited by NOT including classified programs, office equipment such as word processing and calculators, and small expendable munitions, although missiles and torpedoes are included.

DEFINITIONS

AUTOMATED DATA PROCESSING

- COMMERCIAL OFF-THE-SHELF
- "INFORMATION TECHNOLOGY"
- GENERAL PURPOSE
- ACQUISITION BY GSA UNDER BROOKS LAW

EMBEDDED

- SPECIALLY DESIGNED (eg MIL-SPEC)
- ACQUIRED AS PART OF WEAPONS PACKAGE

There are a few terms used in this presentation that need to be defined. "ADP" and "Embedded" are two terms in DOD which have several definitions. For purposes of this report, we have used the simplistic definitions as follows: ADP is characterized by computers generally thought of as commercial - off the shelf. There is a new term "information technology" which was coined by the President's Federal ADP Reorganization Project, and is synonymous with ADP. ADP usually conjures thoughts of computers that are "general purpose." And ADP products are usually acquired by the GSA functioning under the Brooks Law (PL 89-309). "Embedded" computers, on the other hand, are specially designed, for example, designed to meet MIL-Specs, and are acquired as part of a total weapons package, thus "embedded" in a weapon system. Embedded computers are program managed by the 5000.XX series of directives.

HW - COMPUTER HARDWARE RELATED S&S - SOFTWARE & SERVICES e.g.

**SOFTWARE
SYSTEM DESIGN
MAINTENANCE
TRAINING
ETC.**

We use the term "HW" to denote computer hardware-related activities and the term "S&S" for software and services which includes software, system design, maintenance, training and other labor-intensive activities.

**MICRO PROCESSOR – ASSOCIATED WITH "COMPUTER-
ON-A-CHIP TECHNOLOGY**

**MINI PROCESSOR – SMALL COMPUTER, USUALLY
16-BIT, BOARD OR BOX**

**MAXI PROCESSOR – LARGE MINI, USUALLY 32-BIT
WITH MORE MEMORY**

**LARGE SCALE PROCESSOR – CLASSIC BEHEMOTH OF
THE INDUSTRY**

"Microprocessor" refers to computer-on-a-chip technology. "Miniprocessor" is a small computer, usually 16-bit word length, packaged on a board or in a box with associated memories, power supply, etc. A "maxiprocessor" is a more powerful mini, usually 32-bit word length with more memory. Finally, "large-scale processor" is the classic behemoth of the industry such as the IBM 370 or CDC CYBER class of computers.

SOURCE OF INFORMATION

- DOD BUDGET DATA
- CONGRESSIONAL TESTIMONIES
- PERSONAL INTERVIEWS (INDUSTRY, DOD,
CONGRESS STAFF, OMB, GSA)
- PERIODICALS
- INDUSTRY MARKET RESEARCH (F&S, DMS,
QUANTUM, ET AL)
- PUBLISHED GOVERNMENT DATA
(OMB, GSA, GAO, ET AL)

The study team used multiple sources of information including DOD Budget Data; Congressional Testimonies; more than 40 personal interviews with experts in industry, DOD, Congressional staff, OMB and GSA; periodicals, industry market research publications from F&S, DMS, Quantum, et al and published data from several government sources including OMB, GSA, GAO, et al.

**PRESIDENT'S REORGANIZATION PROJECT-
FEDERAL DATA PROCESSING
REORGANIZATION PROJECT COMPLETED
APRIL 1979-11 VOLUMES AVAILABLE
FROM NTIS**

One such published document is the President's Reorganization project completed in April 1979. There are eleven volumes in the final report and these are available from NTIS. I would like to share a few quotations from this report as a preface to our study:

"THE FEDERAL GOVERNMENT IS IRREVERSIBLY
AND INCREASINGLY COMMITTED TO THE USE OF
INFORMATION TECHNOLOGY TO MANAGE ITS RE-
SOURCES, PROVIDE ITS SERVICES, AND PROTECT
ITS CITIZENS."

". . . AN URGENT NEED TO EXPLOIT AND ACCELERATE THE
APPLICATION AND DEVELOPMENT OF INFORMATION TECH-
NOLOGY TO REDUCE THE COST OF GOVERNMENT, IMPROVE
SERVICE DELIVERY, PROTECT OUR PRIVACY, IMPROVE OUR
INDIVIDUAL AND MILITARY SECURITY, AND MAINTAIN WORLD
LEADERSHIP IN A TECHNOLOGY THAT HOLDS THE KEYS TO A
NEW ERA."

"THE FEDERAL GOVERNMENT IS, IN GENERAL, MISMANAGING ITS INFORMATION TECHNOLOGY RESOURCES AND HAS NOT DEVELOPED A PLAN FOR EXPLOITING THE OPPORTUNITIES OF THE FUTURE WITH RESPECT TO INVESTMENT, SERVICE DELIVERY, PROTECTION OF CITIZENS, OR NATIONAL SECURITY."

"INFORMATION TECHNOLOGY IS OF GREAT IMPORTANCE TO THE DOD MISSION. NEARLY ALL MISSION-ESSENTIAL DOD OPERATIONAL AND MANAGEMENT PROCESSES ARE NOW DEPENDENT UPON INFORMATION TECHNOLOGY, AND ENORMOUS RESOURCES ARE EXPENDED ON THEM."

**"THE TOTAL DOLLAR VALUE OF THE
COMPUTER RESOURCES WITHIN DOD
IS UNKNOWN."**

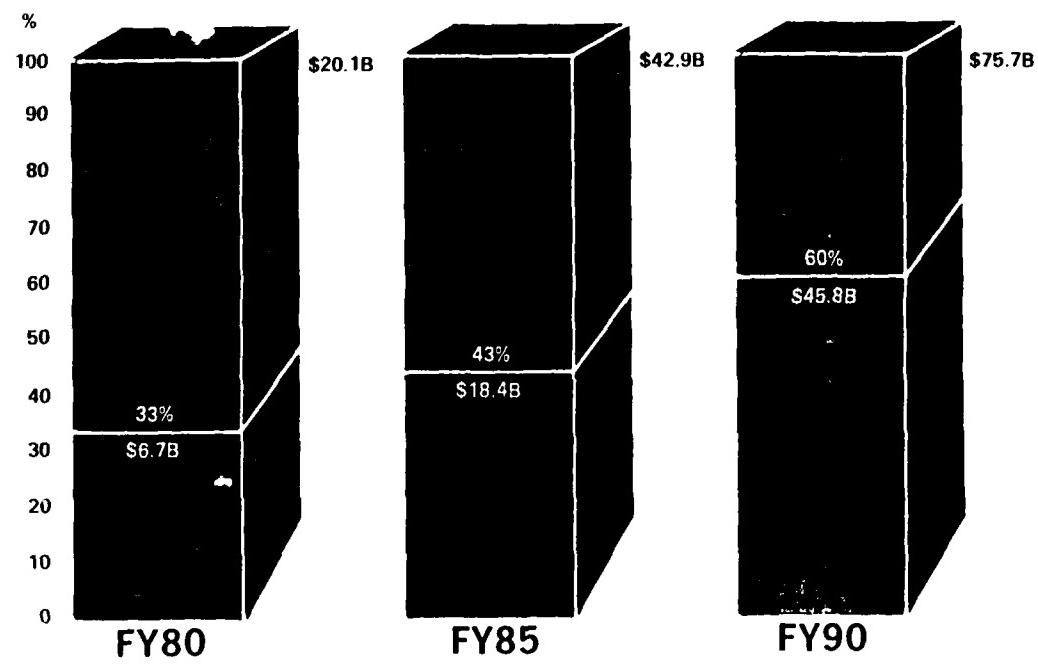
"The total dollar value of the computer resources within DOD is unknown." This then is the springboard into the results of our study. The next chart is in your handout and we will comment only briefly on it, because the following charts will graphically depict the contents.

DOD BUDGET FORECAST (\$BILLIONS)

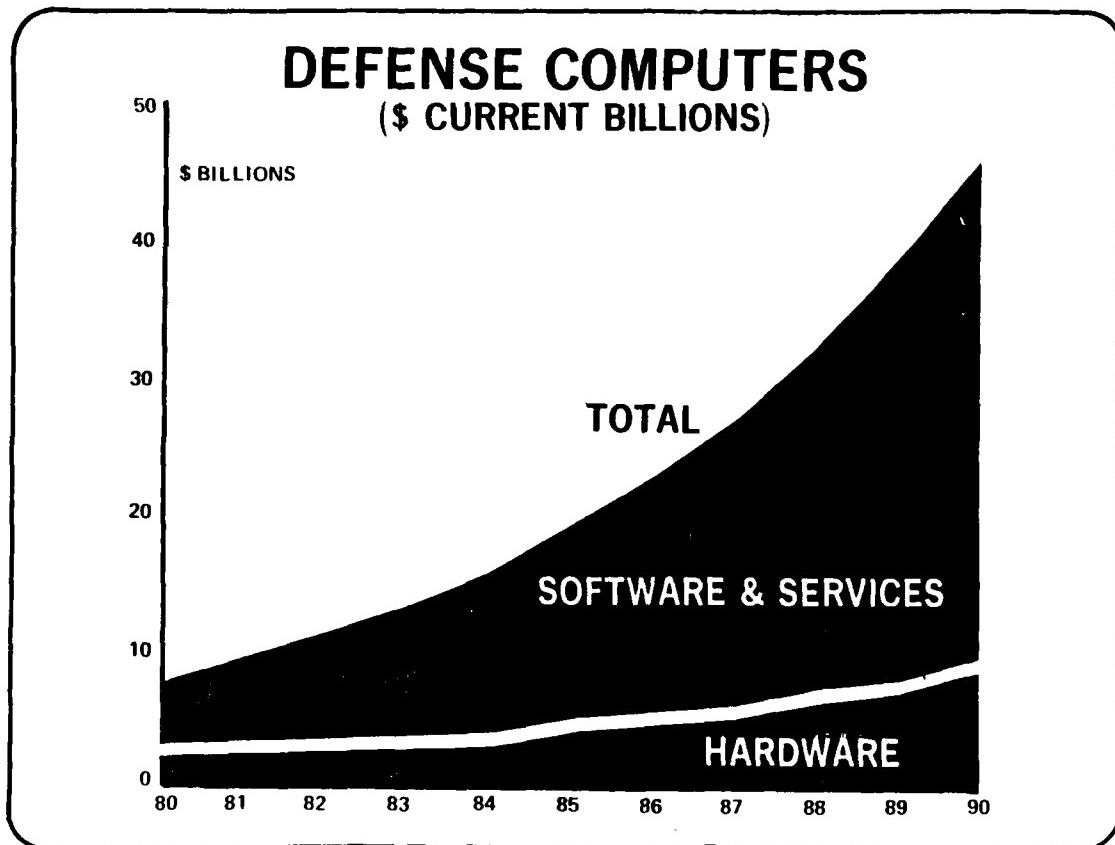
	FY	80	81	82	83	84	85	86	87	88	89	90
TOTAL DOD CURRENT \$		141.7	167.0	190.4	214.2	236.8	260.8	285.2	311.2	339.3	369.3	401.4
DEFLATOR		90.9	100.0	109.0	117.7	126.5	135.7	145.2	155.3	166.2	177.8	190.2
TOTAL DOD 1981 \$		155.9	167.0	174.8	182.0	187.2	192.2	196.5	200.4	204.1	207.7	211.1
DEFENSE ELECTRONICS (TOA)												
CURRENT \$		20.1	25.4	29.6	34.0	38.5	42.9	47.9	53.9	60.0	67.6	75.7
1981 \$		22.1	25.4	27.2	28.9	30.4	31.6	33.0	34.7	36.1	38.0	39.8
DEFENSE COMPUTER												
CURRENT \$		6.7	8.9	10.6	12.8	15.3	18.4	22.1	26.5	31.8	38.2	45.8
1981 \$		7.4	8.9	9.7	10.9	12.1	13.6	15.2	17.1	19.1	21.5	24.1
% OF ELECTRONICS		33%	35%	36%	38%	40%	43%	46%	49%	53%	56%	60%
COMPUTER CONTENT												
HW		31%	29%	27%	26%	24%	23%	22%	21%	20%	19%	19%
CURRENT \$		2.1	2.6	2.9	3.3	3.7	4.3	4.9	5.6	6.5	7.4	8.6
S & S		69%	71%	73%	74%	76%	77%	78%	79%	80%	81%	81%
CURRENT \$		4.6	6.3	7.7	9.5	11.6	14.1	17.2	20.9	25.3	30.8	37.2

I would like to point out that the first line is the EIA estimate of where the total DOD budget is going. The second set of numbers is the EIA ten-year forecast for Electronics. The third set of numbers is our estimate of the total DOD computer budget and the bottom of the chart highlights the computer budget by hardware and Software and Services Content. All dollars in this report are current dollars unless otherwise noted.

DIGITAL COMPUTERS AS % OF DEFENSE ELECTRONICS (\$ CURRENT BILLIONS)

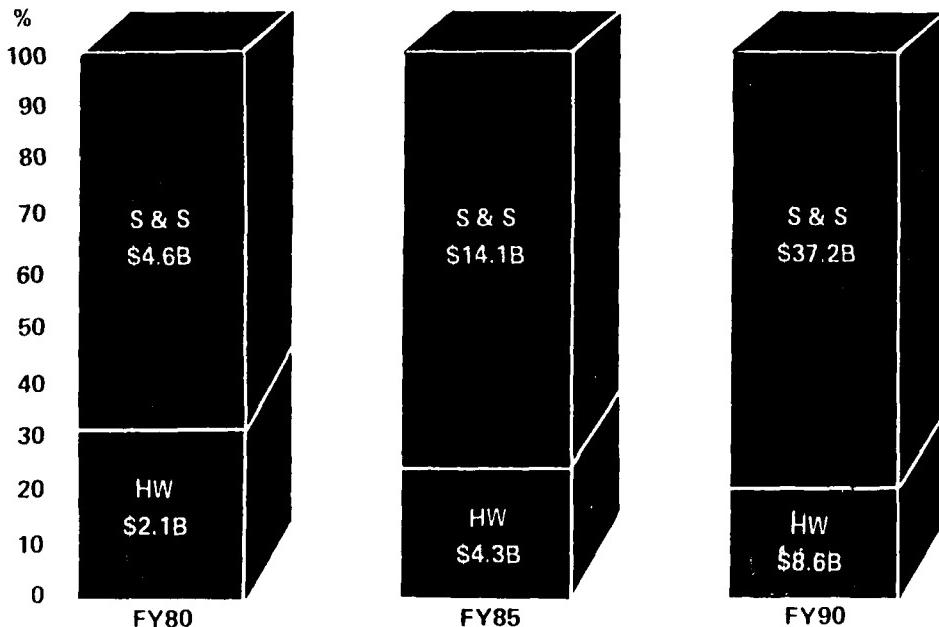


It appears that "DOD Digital Computers" and Defense Electronics are becoming synonomous. As shown here, computers are 33 percent of Electronics in 1980, 43 percent in 1985 and 60 percent in 1990.



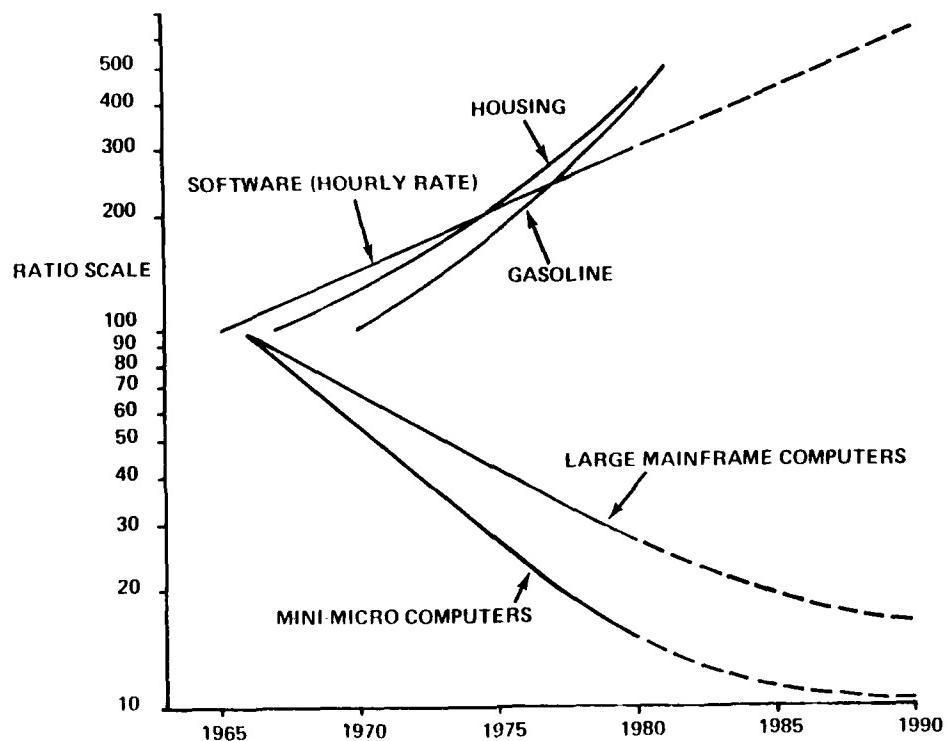
As we might suspect, the Software and Services portion of Defense computers is growing much faster than hardware, growing from \$4.6B (69 percent) in 1980 to \$37.2B or 81 percent of the total in 1990.

DEFENSE COMPUTERS HARDWARE vs SOFTWARE & SERVICES



Or shown differently on this chart, S&S is rapidly on the path to becoming the most significant portion of the DOD computer budget. What are some of the underlying factors causing this development?

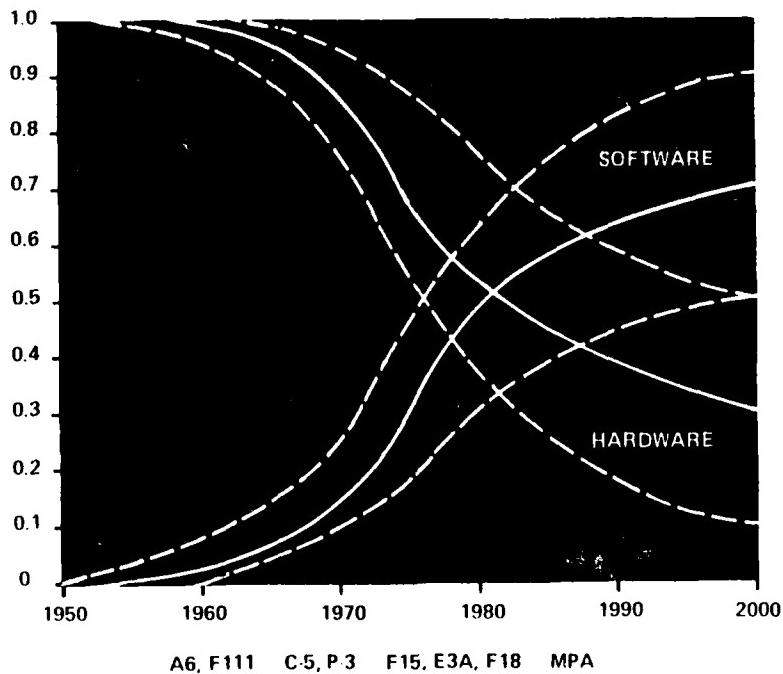
COST TRENDS



Plotted here on a logarithmic ratio scale are several cost trends we can all relate to. The price of gasoline and housing in the past ten years. Hardware decreases for both large mainframe computers and more significantly, in mini and microcomputers. And, of course, the cost of labor which directly impacts labor-intensive activities such as software shown here.

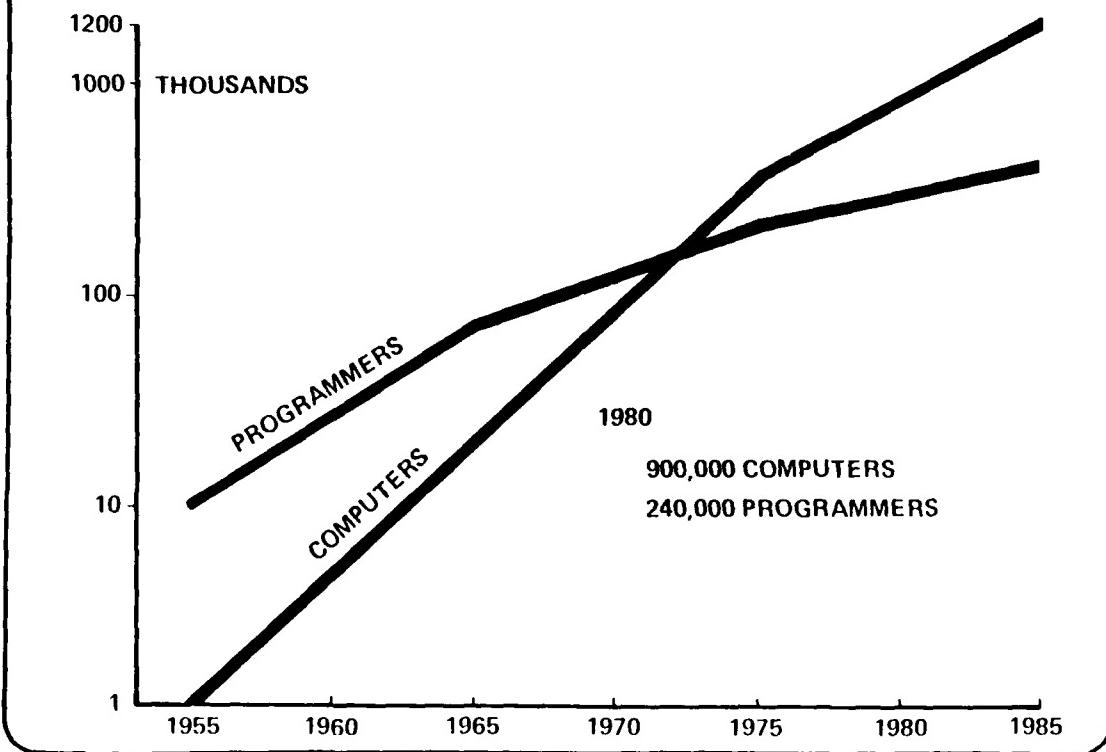
SOFTWARE IMPACT

RELATIVE
INFLUENCE
ON SYSTEM
DESIGN AND
DEVELOPMENT



In fact, the cost of software now exceeds the cost of hardware in most ADP and embedded systems. As shown here, this trend is likely to continue.

COMPUTERS & PROGRAMMERS IN U.S.



Another factor is the shortage of computer programmers. There were approximately 1000 computers and 10,000 programmers in 1955; by 1980 there are approximately 900,000 computers installed in the U.S. with only 240,000 programmers. We predict that the shortfall in programmers will become worse and create additional pressure to the spiraling cost of software and, of course, the Federal government and DOD are vying for the same software resources as industry.

DURING THE '80's

DOD BUDGET INCREASES 2.8 TIMES

DOD ELECTRONICS INCREASE 3.8 TIMES

DOD COMPUTERS INCREASE 6.8 TIMES

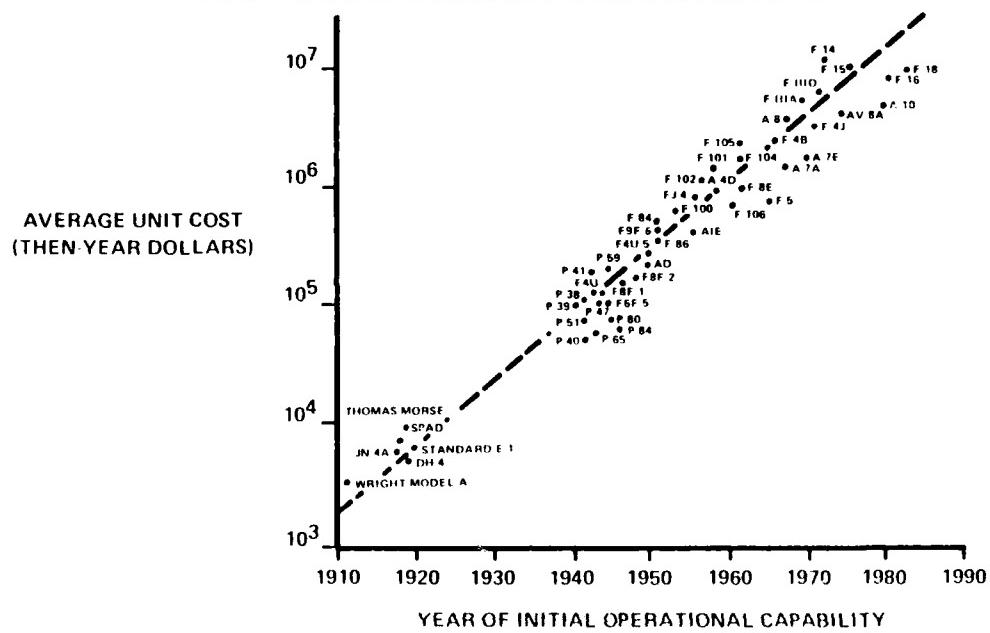
DOD SOFTWARE INCREASES 8.1 TIMES

To summarize this data, during the 80's:

- The DOD budget will increase by a factor of 2.8,
- The DOD Electronics portion will increase by a factor of 3.8,
- DOD computer costs will increase by a factor of 6.8 and,
- DOD software costs will increase by nearly an order of magnitude (8.1 times)!

AUGUSTINE'S LAW NUMBER VIII

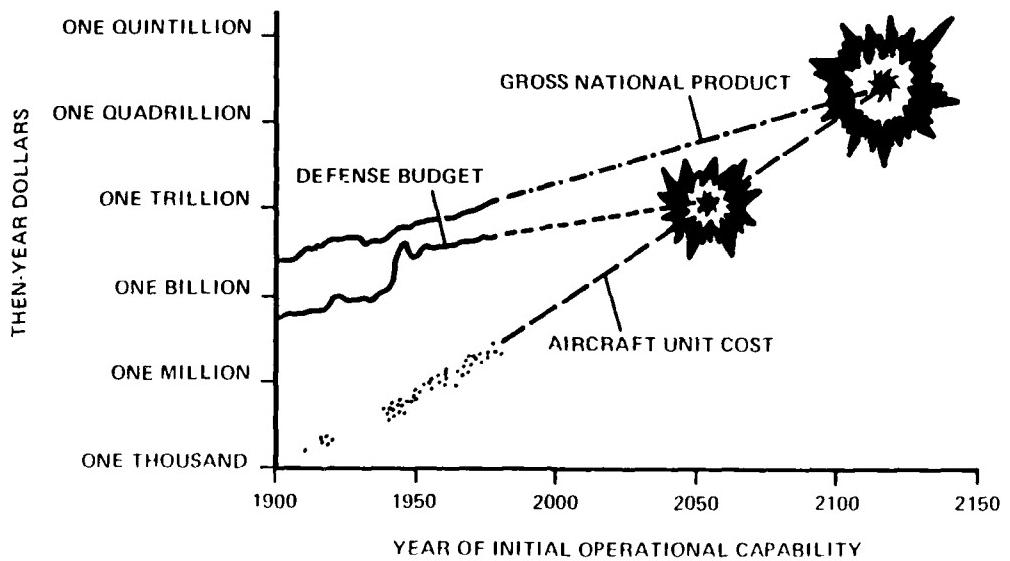
TREND OF INCREASING COST OF TACTICAL AIRCRAFT



Most of you have probably seen Augustine's Law Number VIII.

Plotted here is the then-year cost of various aircraft - from a few thousand dollars for the Wright Model A to the \$20 million per unit for current aircraft.

AUGUSTINE'S LAW NUMBER VIII CALVIN COOLIDGE'S REVENGE



Extending these costs trends, one can conclude that the cost of a single aircraft by the year 2050 will consume the entire DOD budget, and by the year 2130 it will take the entire GNP to purchase one aircraft! Unfortunately, there's an analogy to computer software that looks like this.

STEPHAN'S COROLLARY TO AUGUSTINE'S LAW NUMBER VIII

**IF AIRCRAFT UNIT COST INCREASES ONE
ORDER OF MAGNITUDE EVERY 20 YEARS,
AND SOFTWARE COSTS INCREASE ONE
ORDER OF MAGNITUDE EVERY 10 YEARS,
THEN BY YEAR 2015, SOFTWARE WILL CONSUME
THE ENTIRE DEFENSE BUDGET.**

If aircraft unit cost increases one order of magnitude every 20 years, and software costs increase one order of magnitude every 10 years, then by year 2015, software will consume the entire defense budget! And there won't be any money for aircraft, or tanks, or ships...

DEFENSE ADP FORECAST

Let's now examine the ADP and Embedded markets in greater detail.

Let's look first at the ADP market.

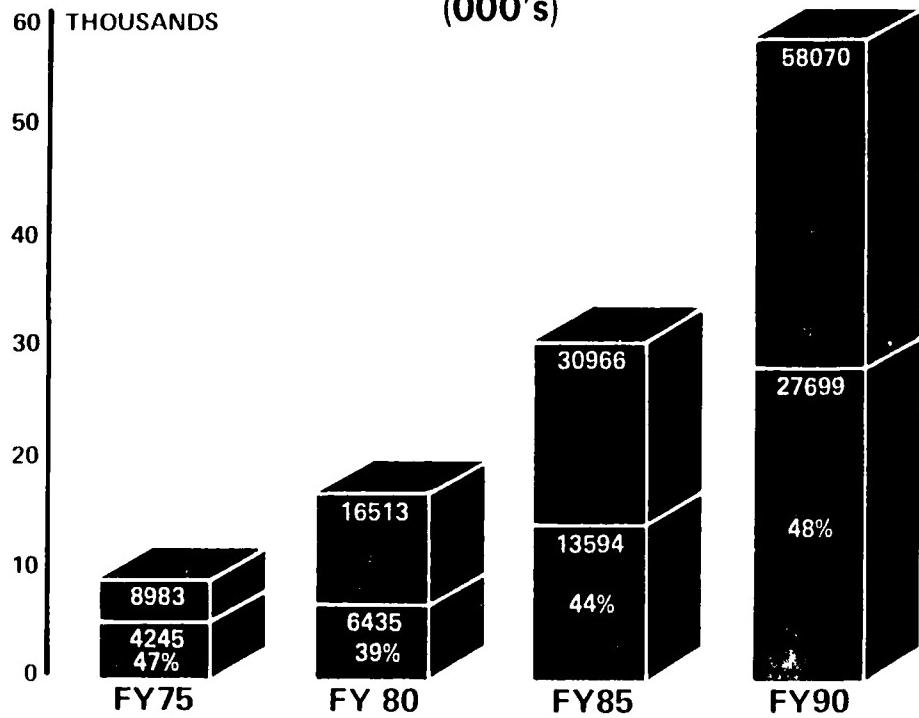
FEDERAL AND DOD AUTOMATIC DATA PROCESSING ADP (\$ BILLIONS)

FY	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
FED ADP BUDGET	3.10	3.29	3.75	4.12	4.77	5.30	5.76	6.43	7.19	8.03	8.97	10.0	11.19	12.50	13.96	15.59
% GROWTH / YEAR	16.5	6.1	14.0	9.8	15.8	11.1	8.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7
# CPU'S	8983	9878	11518	13181	14984	16513	18725	21234	24080	27307	30966	35115	39821	45157	51208	58070
% GROWTH / YEAR	15.2	10.0	16.6	14.4	13.8	10.2	13.4	13.4	13.4	13.4	13.4	13.4	13.4	13.4	13.4	13.4
DOD ADP BUDGET	1.52	1.55	1.91	1.93	2.31	2.60	2.83	3.17	3.56	3.99	4.48	5.00	5.61	6.26	6.99	7.81
% GROWTH / YEAR	8.6	2.0	23.2	1.0	19.7	12.6	8.8	12.0	12.3	12.1	12.3	11.6	12.2	11.6	11.7	11.7
AS % OF FED. ADP BUD	49.0	47.1	50.9	46.8	48.4	49.0	49.1	49.3	49.5	49.7	49.9	50.0	50.1	50.1	50.1	50.1
# CPU'S	4245	4425	5059	5513	6306	6435	7072	8281	10137	11414	13594	15696	18118	20907	24118	27699
% GROWTH / YEAR	5.9	10.0	14.3	9.0	14.4	2.0	9.9	17.0	22.4	12.5	19.0	15.5	15.4	15.3	15.4	14.8
AS % OF FED # CPU'S	47.3	44.7	43.9	41.8	42.1	39.0	37.8	39.0	42.1	41.8	43.9	44.7	45.5	46.3	47.1	47.7
HW	-	-	-	-	-	84	99	108	121	136	152	170	191	213	238	266
S & S	-	-	-	-	-	1.76	1.85	2.09	2.35	2.63	2.96	3.30	3.70	4.13	4.61	5.15

HW - HARDWARE
S & S - SOFTWARE & SERVICES

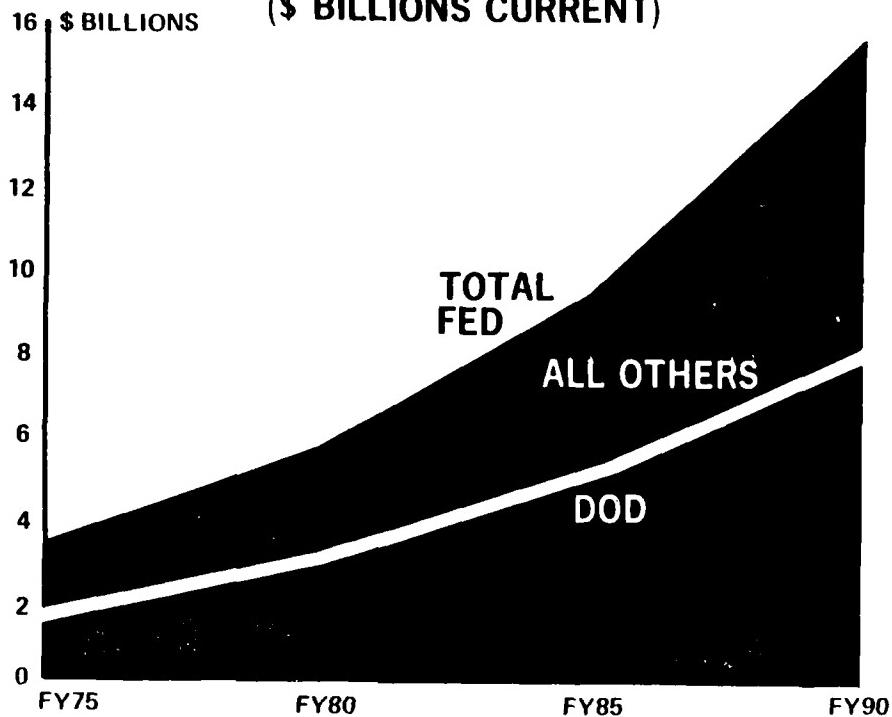
This is another "busy" table of data, included in your handout. The following charts will present the salient points of this set of data. Note that the total Federal ADP forecast and the DOD ADP forecast is included here.

FED & DOD # OF CPU's (000's)

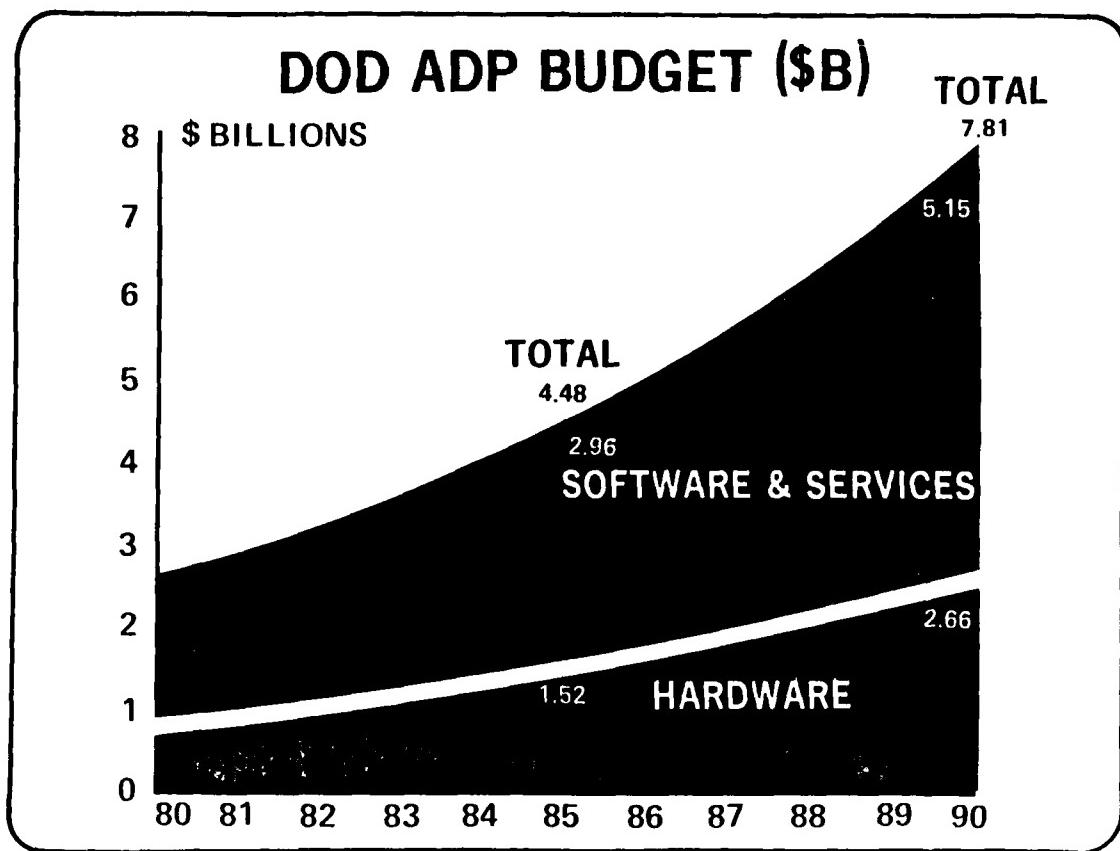


First, the number of computers or CPUs. There were 8983 CPUs in the Federal inventory in 1975 with 47 percent of 4245 belonging to DOD. In the post-Viet Nam era, DOD's ADP inventory increased to 6435 CPUs by 1980, a decline in percent of the total Fed. We are predicting that this trend will reverse and DOD will have 13,594 CPUs in inventory by 1985 and 27,699 CPUs (many of these will be minicomputers) by 1990 which will be about 48 percent of the total in Federal inventory.

FED & DOD ADP BUDGET (\$ BILLIONS CURRENT)



Budget-wise, DOD's ADP is running at about 50 percent of the Federal ADP budget. From \$1.5B in 1975 to \$2.6B in 1980 and our forecast calls for a continuation of the 50 percent trend for a DOD budget of \$4.5B in 1985 and \$7.8B in 1990.



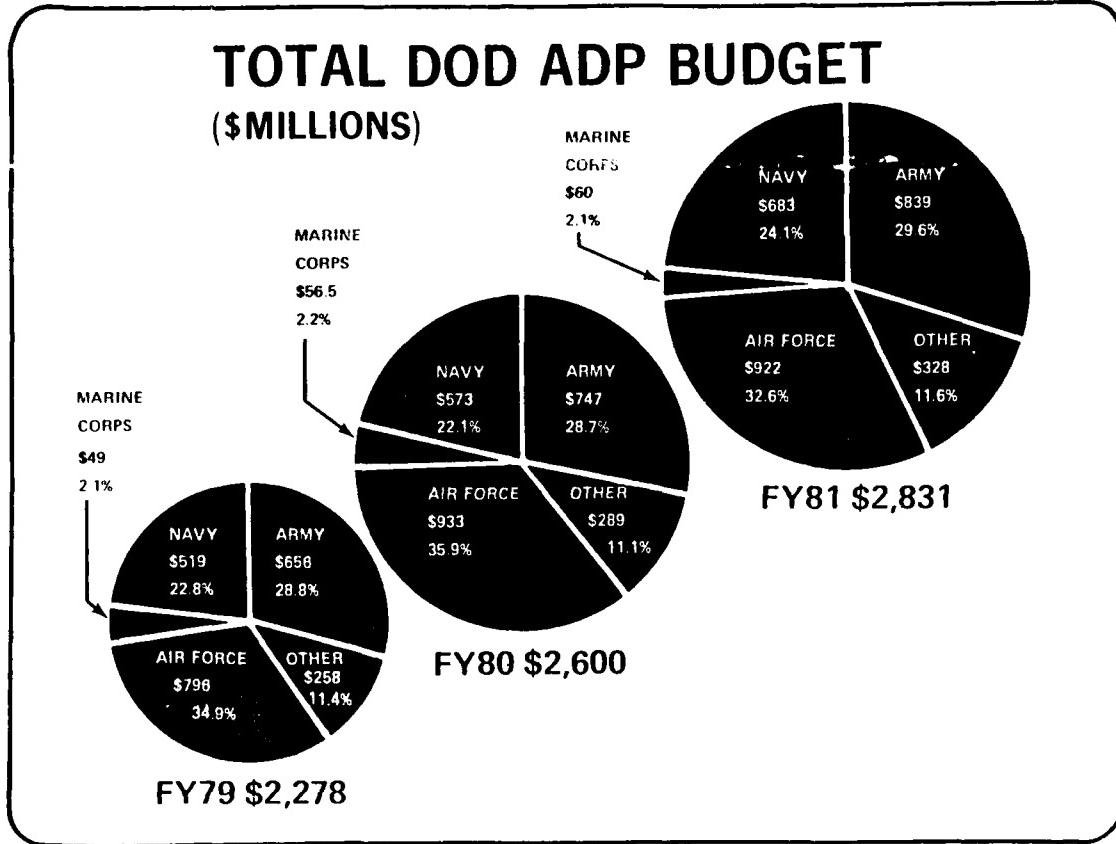
The DOD ADP budget forecast is shown on this chart. Hardware is forecast to increase to \$1.5B by 1985 and to \$2.7B by 1990, but S&S will increase more rapidly to nearly \$3B in 1985 and to over \$5B by 1990.

DOD ADP BUDGET (\$ MILLIONS)

	FY 80 S & S	FY 80 HW	FY 81 S & S	FY 81 HW
I. CAPITAL INVEST. (PROCUREMENT)				
A. PURCHASE OF NEW CAPACITY		206.8		208.7
B. PURCHASE TO EXPAND/REPLACE EXISTING CAPACITY		108.1		146.7
C. PURCHASE OF SOFTWARE	11.4		14.0	
D. SITE CONSTRUCTION		20.3		29.2
II. OPERATIONS (O & M), (RDT&E)				
A. PERSONNEL (57,000 PEOPLE)	1,146.6		1,162.5	
B. EQUIPMENT RENTAL, ETC.				
1. ADPE RENTALS		387.0		457.8
2. SPACE		9.4		9.3
3. SUPPLIES		129.2		145.3
C. COMMERCIAL SERVICES				
1. ADPE TIME	33.8		34.3	
2. OPERATIONS	31.9		38.6	
3. SYSTEM ANAL. & PROG.	250.6		262.2	
4. ADPE MAINTENANCE	166.3		186.2	
5. ADP STUDIES	114.5		156.6	
TOTALS	1,755.1 (67%)	860.8 (33%)	1,854.4 (65%)	997.0 (35%)
HW - HARDWARE RELATED		2,615.9		2,851.4
S & S - SOFTWARE & SERVICES RELATED				

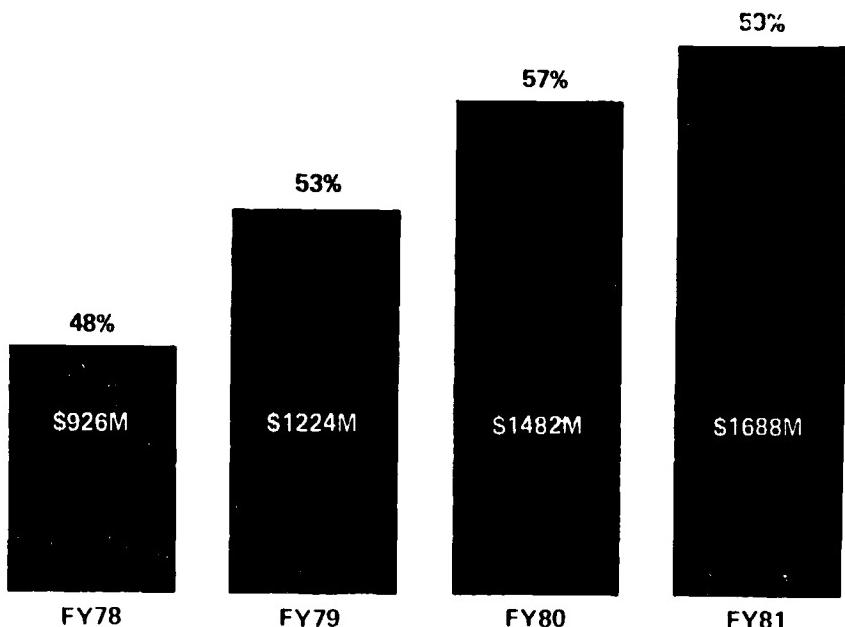
This chart shows the DOD ADP budgets for FY80 and FY81 by category. Capital Investment (procurement funding) includes purchase of new capacity, purchase to expand/replace, purchase of software and site construction. The preponderance of funding is for operations (with most of the funding coming from O&M) and operations includes personnel (constant at about 57,000 people), equipment rental and commercial services. The S&S versus HW content for FY80 and FY81 is indicated.

TOTAL DOD ADP BUDGET (\$MILLIONS)



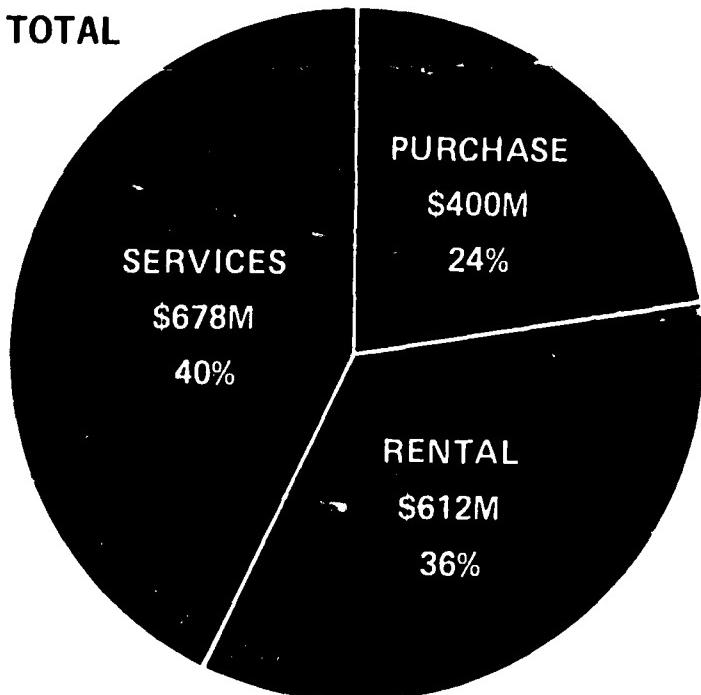
The split between services is indicated here for FY79, 80 and 81. Navy and Army ADP budgets are growing percentage-wise while Air Force ADP is showing a slight decrease percentage-wise.

DOD ADP \$ TO PRIVATE SECTOR



The ever-increasing DOD ADP budget combined with nearly constant in-house personnel levels is reflected here. Forty-eight (48) percent of the FY78 budget went to industry whereas 59 percent (\$1688M) of the FY81 budget will go to industry.

FY81 PRIVATE SECTOR PORTION OF ADP
\$1,688M TOTAL



The \$1.7B consists of \$400M (24 percent) for purchase, \$612M (36 percent) for rental and \$678M (40 percent) for services. "Services" include ADPE time, operations, system analysis and programming, ADPE maintenance, and ADP studies.

SOFTWARE TRENDS

- MORE LINES OF CODE + MORE MAINTENANCE
X LABOR SHORTFALL = RAPID INCREASE IN COSTS
- STANDARDIZATION: EFFORTS TO ACCELERATE
 1. TOOLS eg Ada HOL
 2. MODULES eg REUSE OF PACKAGES
- BUT ... DOD VERIFICATION/CERTIFICATION (QA)
WILL ADD TO COSTS UNTIL LATE '80'S
- NATURAL LANGUAGE EXTENSION TO HOL
BY END OF '80's (VOICE INPUT/ OUTPUT)
- CAD/CAM – LIKE SOFTWARE PRODUCTION
FROM SYSTEM DESIGN SPECIFICATION

I would like to conclude the ADP portion of this presentation with a few predictions and trends which seem apparent. In software, more computers mean more lines of code; this fact plus more maintenance costs times the programmer shortfall is going to continue to force software costs up. Currently software costs about \$50 per line of code, and nearly two-thirds of software expenditures go for maintenance. To counter this trend, there will be new efforts placed on standardization - in two primary initiatives:

1. There will be greater emphasis placed on software tools such as the Ada HOL effort aimed at increasing programmer productivity.
2. There will be efforts to standardize on reusable software modules such as operating systems and application packages. Some of these will be "burned" into hardware modules. But, standardization costs money and efforts to standardize won't decrease overall software costs until late in the decade. Look for extensions to the Ada HOL to include natural language by the end of the 80's with use of direct voice recognition as computer I/O. We will see automated production of software evolve during the decade.

HARDWARE TRENDS

- MINI'S AND MICRO'S WILL PROMOTE DECENTRALIZED ARCHITECTURES TIED TO SMART MASTER
- VLSI WILL YIELD GAINS IN RELIABILITY, PERFORMANCE, REDUCED COSTS
- STANDARDIZATION WILL OCCUR AT INTERFACE LEVEL

BUT...

THESE HARDWARE TRENDS WILL ADD FUEL TO THE SOFTWARE FIRE.

In the hardware area: mini's and micro's will promote decentralized architectures tied to smart hosts or masters. VLSI will yield gains in reliability, performance and reduced costs. Standardization will tend to occur at the interface level. But, these hardware trends will only contribute to higher software costs in the near future.

ACQUISITION TRENDS

**FACT: DOD ADP CPU'S ARE 12% LEASED
88% OWNED.**

**FACT: AVERAGE AGE OF CPU'S IS 7-9 YEARS
AND INCREASING.**

**HIGH COST OF MAINTAINING AGING
INVENTORY WILL TREND TOWARD MORE
LEASING WHERE LIFE CYCLE COSTS JUSTIFY.**

DOD currently owns 88 percent of its ADP inventory and leases 12 percent. The average age of the inventory is seven to nine years and increasing (about six years older than the private sector). The high cost of maintaining this aging inventory will force DOD to lease new equipment where total life cycle costs justify. Lowest Total Overall Costs (LTOC) will be emphasized with recognition for the large software conversion costs.

• ADP = 10% SOLE SOURCE 90% COMPETITIVE BID

- 1. GREATER PRESSURE TO
COMPETE BIDS**
- 2. GSA WILL ISSUE FEWER
DELEGATION OF PROCUREMENT
AUTHORITIES (DPA'S)**
 - THRESHOLD WILL BE RAISED**
 - PRESSURE TO COMPETE**

Purportedly, 90 percent of Federal ADP procurements are competitive. This percentage is much smaller in DOD for a variety of reasons. There will, however, be increased pressure to acquire ADP products on a competitive basis and GSA will issue fewer waivers (or DPA's) for two reasons: the acquisition threshold will probably be increased from \$300,000 to \$500,000 or higher, and Congress will apply pressure to compete. Currently any sole source procurement or any competitive purchase over \$300,000 goes through GSA unless a DPA is issued. There were 346 DPA's issued in 1979: the average competitive was \$3.6M and the average sole source was \$700,000.

PRESIDENT'S REORGANIZATION PROJECT —
(COMPLETED APRIL '79) NATIONAL SECURITY
TEAM RECOMMENDATION WILL BE IMPLEMENTED:

A NEW OFFICE OF INFORMATION TECHNOLOGY
WILL BE CREATED AT HIGH LEVEL IN DOD:

- STAFF & POLICY ROLE FOR ADP &
EMBEDDED AREAS
- DOD FINANCIAL OVERVIEW
- DEAL WITH CONGRESS

Finally, the last point in the ADP portion of this presentation relates to the previously mentioned President's Reorganization Project. The National Security team, in this project, recommended that DOD should create a new high-level organization to deal with information technology (both ADP and Embedded). We believe that this office will be established at the OSD level with responsibility for:

- Staff and policy for both ADP and Embedded.
- DOD financial overview.
- DOD computer dealings with Congress.

DEFENSE EMBEDDED COMPUTER FORECAST

Now, let's take a closer look at the DOD Embedded computer area. The forecast addresses the U.S. military and aerospace market for militarized digital computers which are applied in real-time equipment operations to solve tactical, strategic, and operational problems.

The major computer resource elements are broadly categorized into two groups - hardware and software. . .

Hardware

In this study, the computer is considered to consist of the central processing unit (CPU), the input/output unit, and the main memory unit. The computer, embedded within each system surveyed, is basically a device that is capable of accepting information, applying prescribed processes to the information, and providing the results of these processes.

Software

The computer software resources are functional support software which provides direct support to major software activities, operating system services, post-deployment support software, and applications software.

The excluded markets are:

- Classified Programs
- Data Base Management Systems
- Test Equipment
- Training Simulators

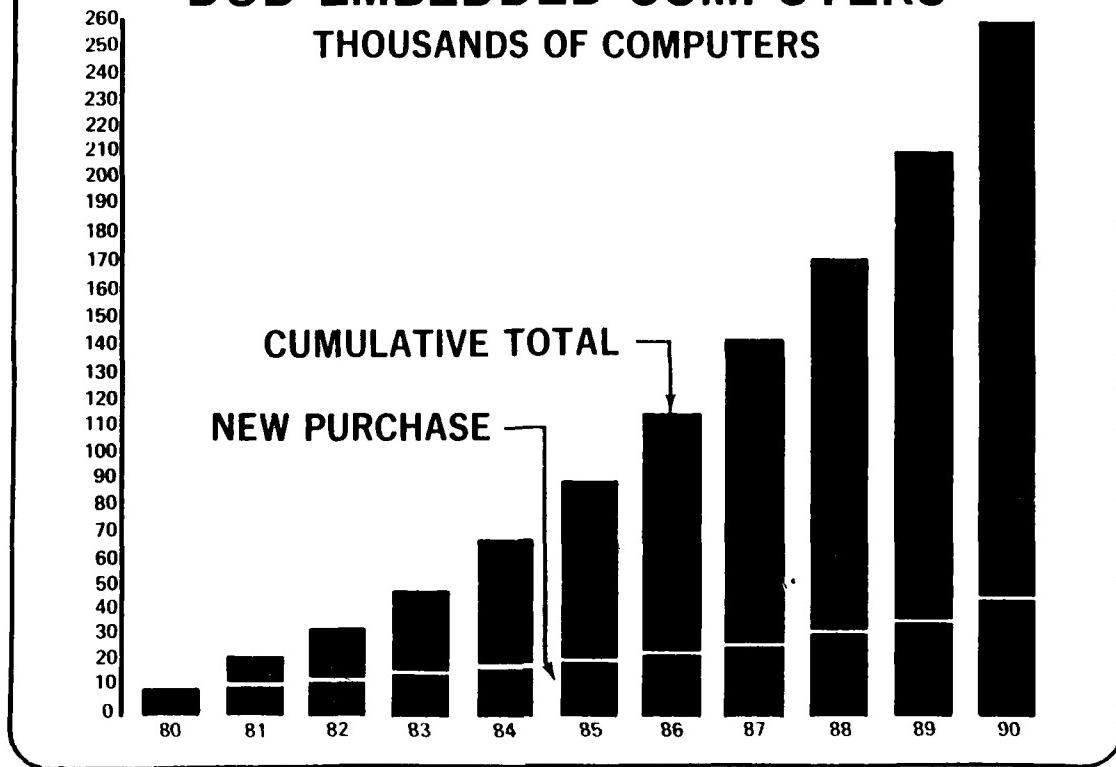
End users in this marketplace are the Army, Air Force, Navy and Marines.

DOD EMBEDDED COMPUTERS

	'80	'81	'82	'83	'84	'85	'86	'87	'88	'89	'90
NO. OF COMPUTERS	10110	11270	12110	14645	18017	20109	23095	26996	31673	37297	44347
CUMULATIVE											
TOTAL	10110	21380	33490	48135	66152	86261	109356	136352	168025	205322	249669
BUDGET (\$ BILLION)											
HARDWARE	1.28	1.58	1.81	2.08	2.36	2.75	3.20	3.73	4.34	5.06	5.89
SOFTWARE	2.82	4.49	5.62	7.16	8.95	11.17	13.90	17.16	21.20	26.15	32.10
TOTAL	4.10	6.07	7.43	9.24	11.31	13.92	17.10	20.89	25.54	31.21	37.99

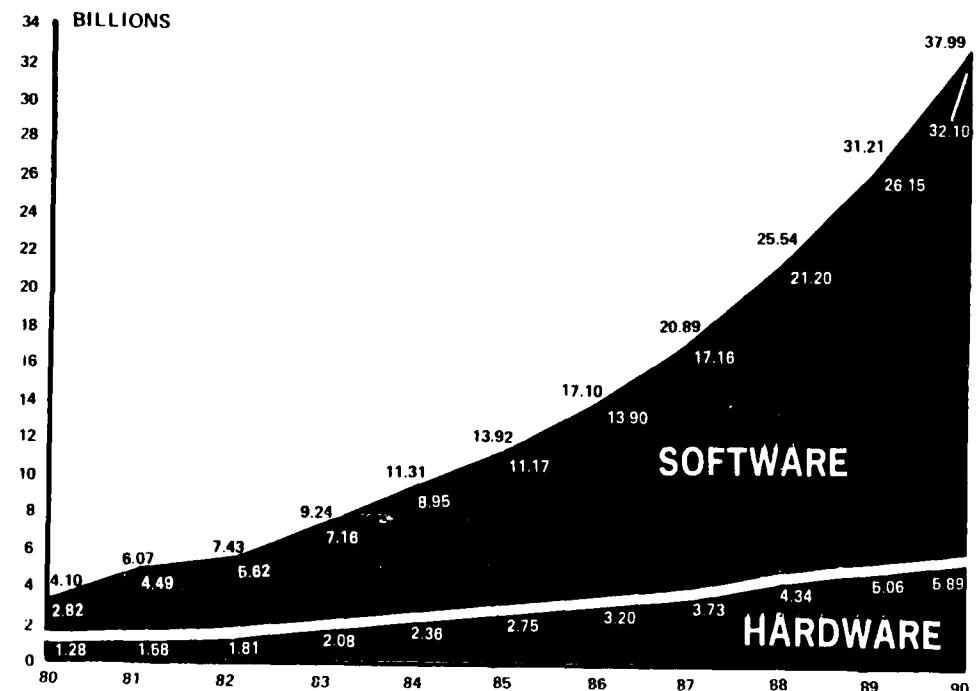
The contents of this table will be graphically presented in succeeding charts. Note that the number of embedded computers is included here as well as the budget which has been subdivided into hardware and software content. The average rate of growth in the number of computers is 16 percent. The average yearly growth for hardware dollars is 17 percent while software has an annual growth rate of 28 percent.

DOD EMBEDDED COMPUTERS THOUSANDS OF COMPUTERS



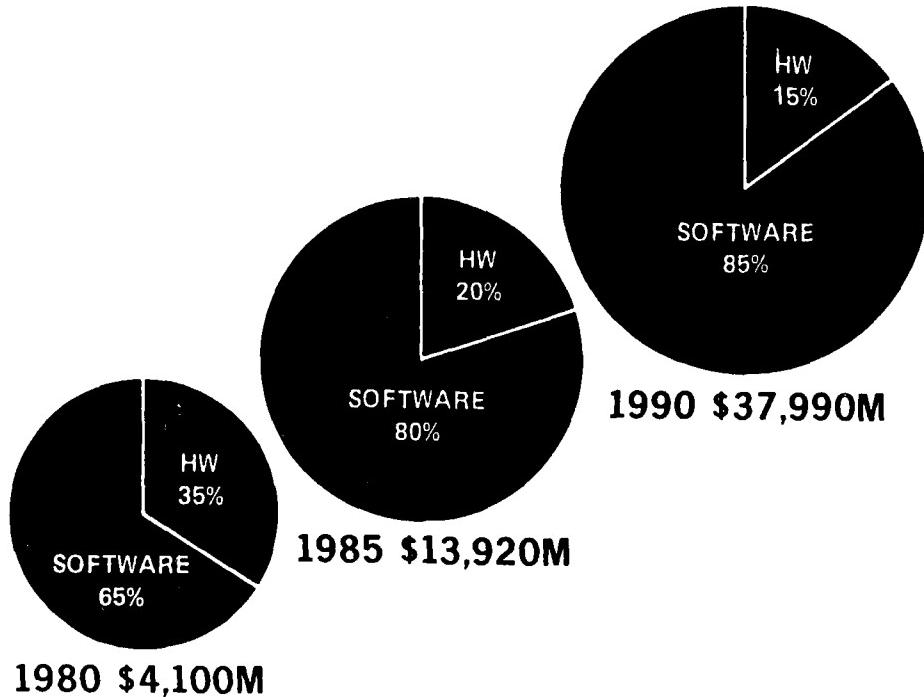
Microprocessors will have an ever-increasing influence in the embedded area - much more so than in the ADP area. As indicated here, the cumulative total of delivered embedded computers will grow from approximately 10,000 in 1980 to about 250,000 in 1990. Nearly every weapon system in the future will have an embedded computer or computers somewhat in its control subsystem or C³I subsystem.

DOD EMBEDDED COMPUTER MARKET SOFTWARE/HARDWARE



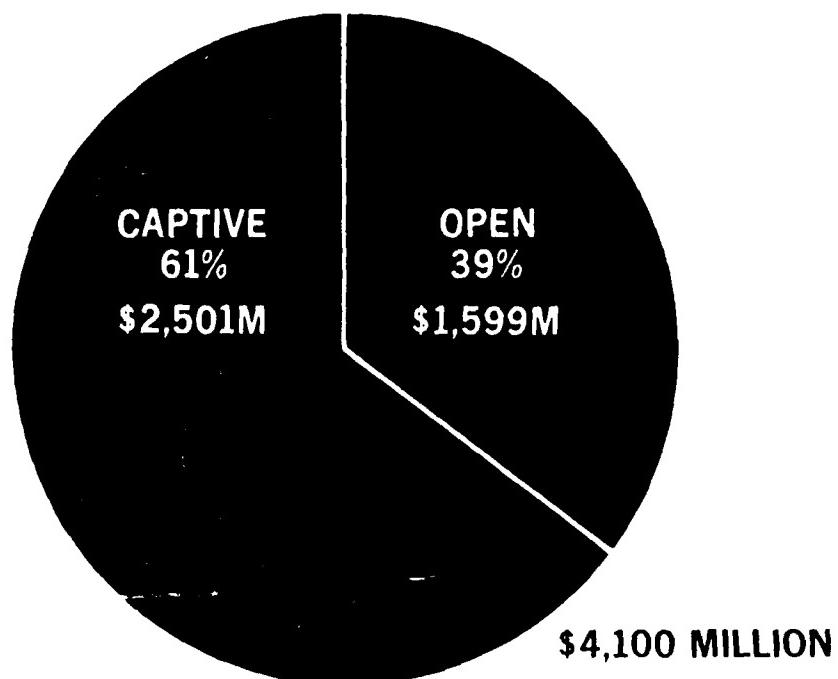
Budget-wise, embedded computers accounted for \$4.1B in 1980 with \$2.82B for software and \$1.28B earmarked for hardware. As you can see, the software content is forecasted to grow to some \$32B by 1990 with hardware increasing, on a smaller scale, to about \$6B. In 1980, software was 69 percent of the total budget; in 1981 software will grow to 71 percent, and our projection shows that software will increase to 85 percent by 1990.

EMBEDDED COMPUTERS HARDWARE vs SOFTWARE



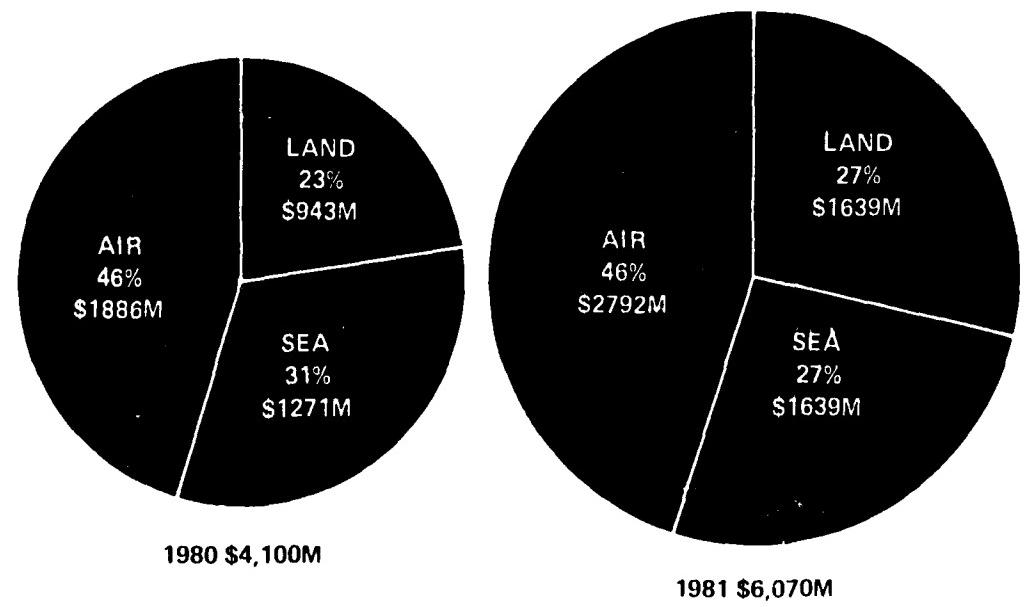
The embedded pie equal to \$4,100M in 1980 consisted of a software piece of 69 percent and a hardware piece of 31 percent. The total pie will grow to nearly \$14B by 1985 with software content increasing to 80 percent, and the pie will grow to \$38B by 1990 with software accounting for 85 percent of the total.

1980 DOD EMBEDDED COMPUTER MARKET CAPTIVE vs OPEN MARKET



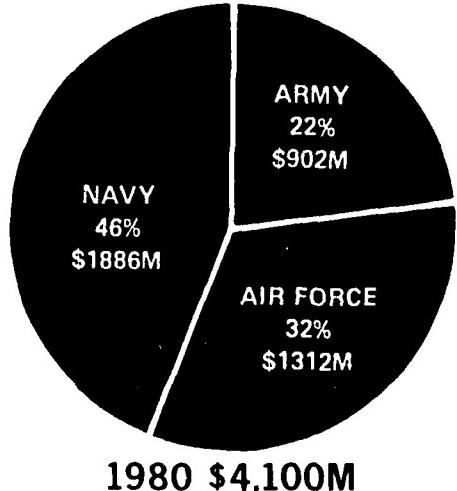
In 1980, 61 percent of the total embedded market was captive with 39 percent open. The "captive" market consists of those programs that have had contract awards or fall under the standard computer programs of a particular service. The "open" market then represents new opportunities for industry. Software has a much larger open market than hardware. The "open" market should increase as technology advances.

EMBEDDED COMPUTERS BY PLATFORM

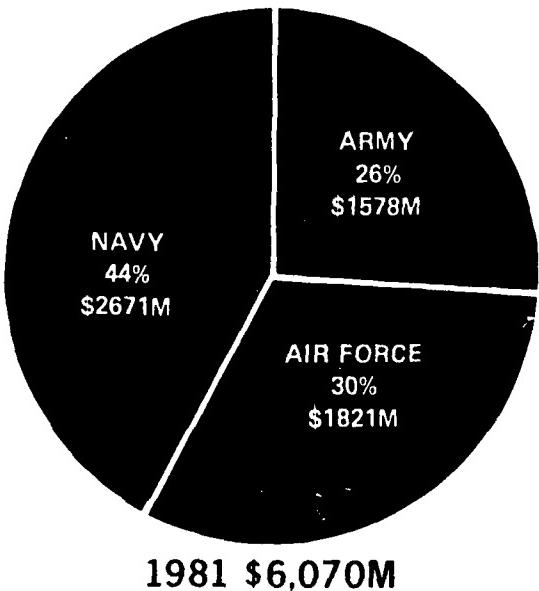


The chart shows the embedded computer budgets for 1980 and 1981 by platform. Airborne platforms account for 46 percent of each budget whereas land-based platforms are growing slightly from 23 percent in 1980 to 27 percent in 1981. Sea, surface and subsurface platforms budget is growing in absolute terms but a slight decline is forecast percentage-wise. Land represents ground-based systems including vehicle mounted systems.

EMBEDDED COMPUTERS BY SERVICE



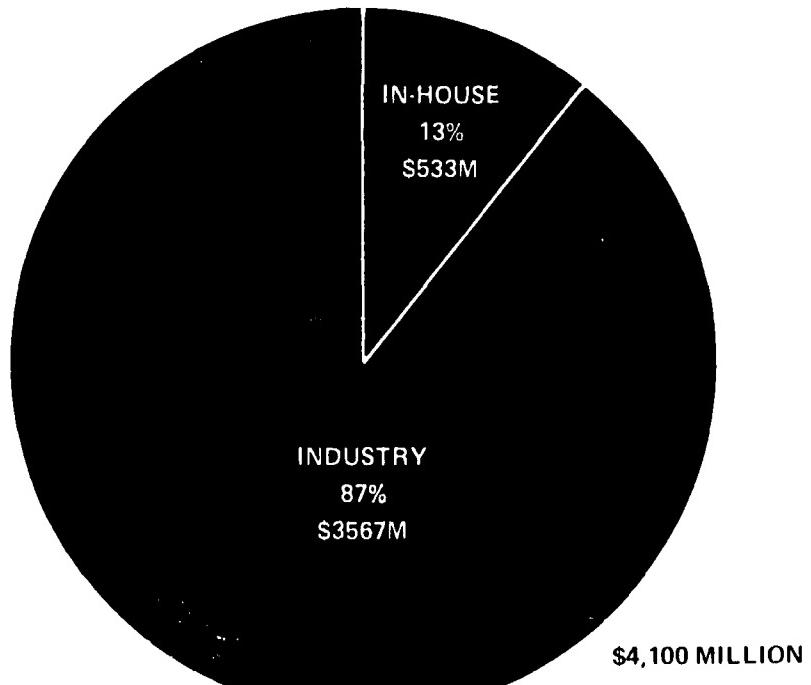
1980 \$4,100M



1981 \$6,070M

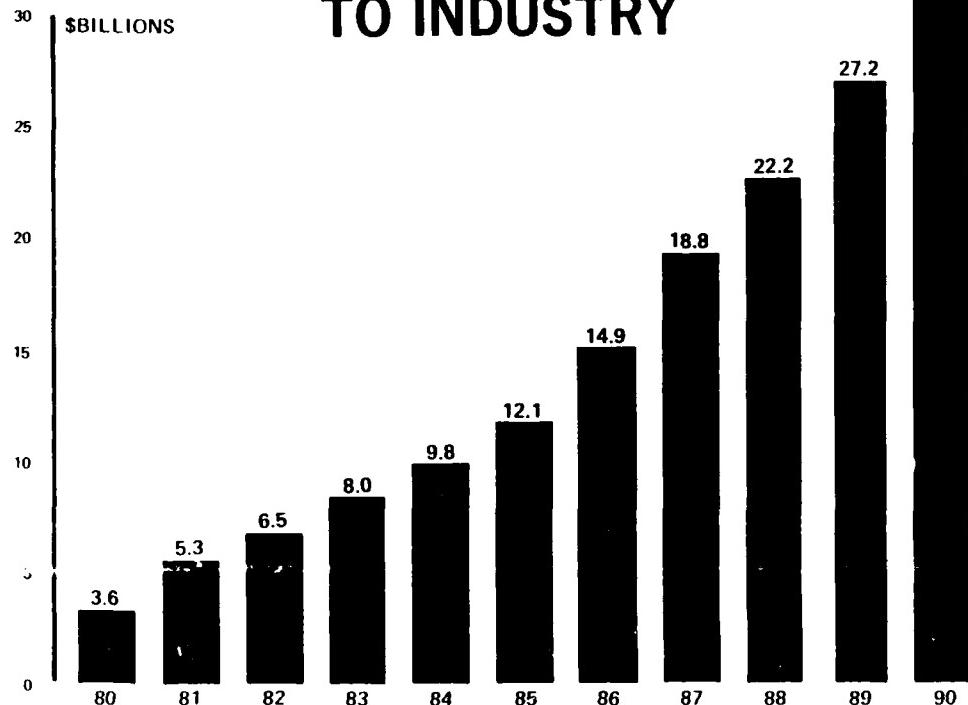
The Navy has the largest budget - 46 percent in 1980 and 44 percent in 1981. The Army is playing catch up with a budget forecast for 1981 of \$1578M (26 percent of the total) up from \$902M (22 percent) in 1980. The Air Force forecast calls for a slight decrease in 1981 percentage-wise.

1980 EMBEDDED DOLLARS RETURNED TO INDUSTRY



A more significant portion of the Embedded budget is returned to industry than from the ADP budget. An estimated 87 percent or \$3567M of the 1980 budget was contracted out. Most of this funding came from RDT&E accounts with a smaller portion from O&M and procurement accounts. There's a definite trend for the services to function more and more as program managers executing contracts to industry in the embedded areas as opposed to performing computer design/development tasks in-house. There are 55 industry suppliers of militarized embedded computers, dominated by five or six major companies, most of whom have a standard computer. The "in-house" portion of the budget is primarily maintenance and software support, data configuration management, logistic support, and post-deployment support.

EMBEDDED DOLLARS RETURNED TO INDUSTRY



Our forecast for the 80's indicates that the embedded computer dollars returned to the industry will increase substantially as virtually every weapon system now in planning or under development will require embedded computers. An ever-increasing percentage of these computers will be microprocessors. In fact, the terms "embedded computer" and "Software" need redefining as we enter the microprocessor era. The personnel shortfall in the services will necessitate even more of the in-house software budget to be returned to industry.

LSI IN MILITARY SYSTEMS

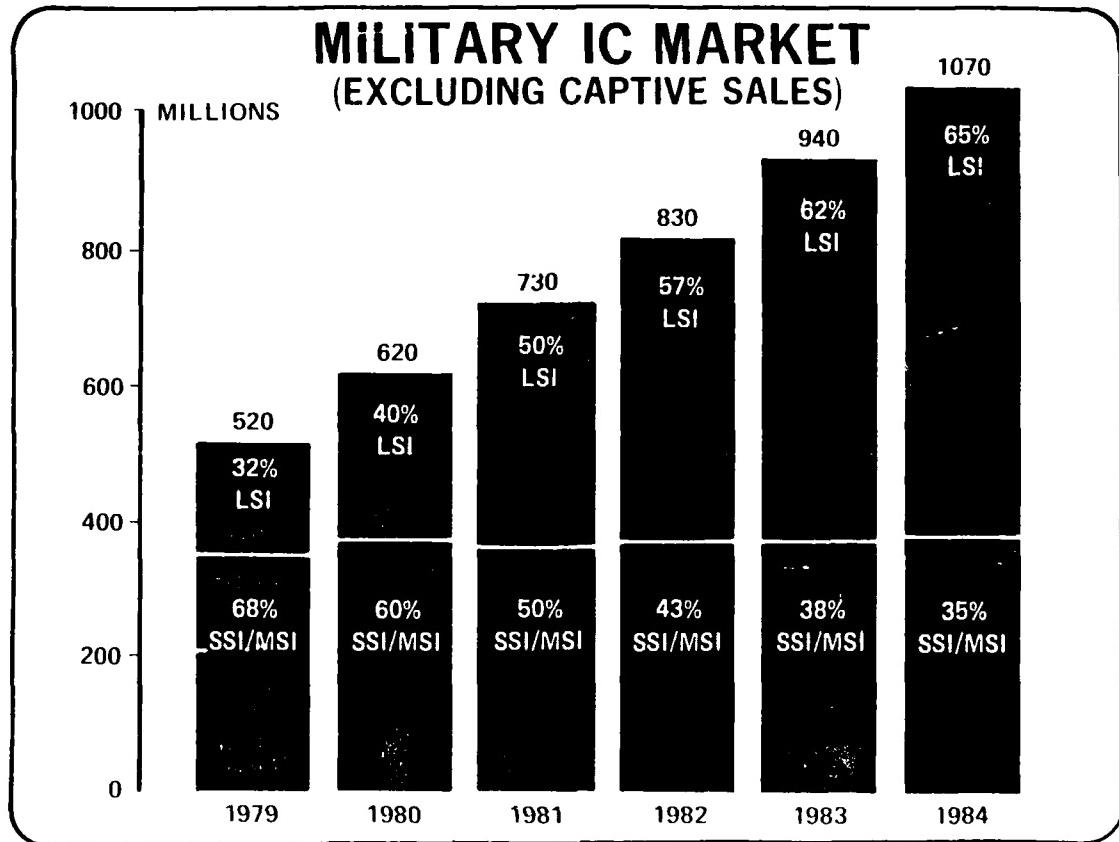
- EMBEDDED COMPUTERS HAVE BECOME AN INTEGRAL PART OF VIRTUALLY EVERY MODERN MILITARY SYSTEM, LARGELY MADE POSSIBLE BY PROGRAMMABLE MICROPROCESSORS
- LARGE SCALE INTEGRATION (LSI) AS TYPIFIED BY MICROPROCESSORS ARE BECOMING DOMINANT IN MILITARY SYSTEMS AT THE EXPENSE OF TRADITIONAL SSI/MSI MICRO CIRCUITS, e.g., TTL
- IN ABOUT 1981 WE CAN EXPECT TO SEE A 50/50 USAGE OF LSI vs SSI/MSI BY DOD WITH LSI DOMINANT THEREAFTER

Large Scale Integration (LSI) is having a significant impact on military systems. Embedded computers have become an integral part of virtually every modern military system, largely made possible by programmable microprocessors. LSI as typified by microprocessors is becoming dominant in military systems at the expense of traditional Small Scale Integration/Medium Scale Integration (SSI/MSI) microcircuits such as Transistor-transistor Logic (TTL). In 1981 or shortly thereafter, we can expect to see a 50 percent usage of LSI and 50 percent SSI/MSI within DOD - with LSI dominant thereafter.

MICROPROCESSOR TYPES IN USE TODAY

- MICROPROCESSORS ARE TAKING OVER FUNCTIONS PREVIOUSLY PERFORMED BY MINI-COMPUTERS
- BY MID-80's VLSI WILL ALLOW MICROPROCESSORS TO OFFER MAINFRAME COMPUTER PERFORMANCE IN MILITARY SYSTEMS
- TWO DISTINCT FORMS OF MICROPROCESSORS ARE IN USE BY MILITARY:
 - SINGLE CHIP CPU (MOS, I²L, CMOS)
 - BIPOLAR BIT-SLICE FAMILIES

Particularly in the embedded area, we see microprocessors taking over functions previously performed by minicomputers. By the mid 80's, Very Large Scale Integration will allow microprocessors to offer mainframe computer performance in military systems. Two distinct forms of microprocessors are in use within military systems today: single chip CPUs using MOS, I²L and CMOS technology and bipolar bit-slice families which are the building blocks of current military compute such as the Navy's standard airborne computer - the AN/AYK-14.



Shown here is the estimated military integrated circuit market in millions of dollars. Note that LSI devices represented 32 percent of the 1979 market and will increase to 50 percent by 1981 and will soar to 65 percent of the \$1 billion forecast for 1984.

FUTURE MICROPROCESSORS

- BY EARLY '80's, WE WILL SEE A SINGLE CHIP MICROPROCESSOR (OR CPU) CAPABLE OF A MILLION INSTRUCTIONS PER SECOND "A MIP ON A CHIP"
- NEXT YEAR INTEL WILL INTRODUCE A MICRO MAIN-FRAME (MULTI-CHIP) 32-BIT MACHINE (THE 432) THAT:
 - PROVIDES MULTI-PROCESSOR PERFORMANCE EQUAL TO IBM 370/158
 - HAS ARCHITECTURE THAT SUPPORTS Ada
 - IS IN A MICRO-COMPUTER FORM FACTOR AND AT A MICRO-COMPUTER COST

Microprocessor technology is changing rapidly. By the early 80's, we will see a single chip microprocessor (or CPU) capable of performing a million instructions per second - "A MIP on a chip."

An example of things to come - next year Intel will introduce a micro mainframe (multiple chips) 32-bit machine (the 432) that will provide multiprocessor performance equal to the IBM 370/158. It will have an architecture designed to support Ada and will be in a microcomputer form factor at a microcomputer cost.

VHSIC PROGRAM

- DURING MID TO LATE '80's, PROGRESS IN LITHOGRAPHY AND PROCESSING WILL ALLOW CHIP DESIGN AT 100,000 GATE LEVEL OF COMPLEXITY OPERATING AT 100 MEGA HERTZ OR FASTER
- VHSIC (6 YEAR \$210 MILLION EFFORT) IS DESIGNED TO MEET DOD NEEDS IN LATE '80's AND BEYOND
- GOALS OF VHSIC PROGRAM:
 - DEVELOP SUBSYSTEMS ON CHIPS USING FABRICATION GEOMETRIES OF 1.25 MICRONS (EVENTUALLY .5 TO .8 MICRONS)
 - EMPHASIS ON INCREASING SYSTEM THROUGHPUT - ULTIMATELY REAL-TIME SIGNAL PROCESSING
 - EXPLOITATION OF HIGH CHIP COMPLEXITY, SUCH AS ON CHIP TEST, FAULT TOLERANCE, ERROR CORRECTION, ETC.
 - ACHIEVE ARCHITECTURAL CONCEPTS TO MINIMIZE NEEDS FOR CUSTOM DESIGNS

A few words on DOD's Very High Speed Integrated Circuit (VHSIC) program. During the mid to late 80's, progress in lithography and processing will allow chip design at 100,000 gate level of complexity operating at 100 megahertz or faster. VHSIC - the six year \$210M effort - is designed to meet DOD needs in the late 80's and beyond. Goals of the VHSIC program include:

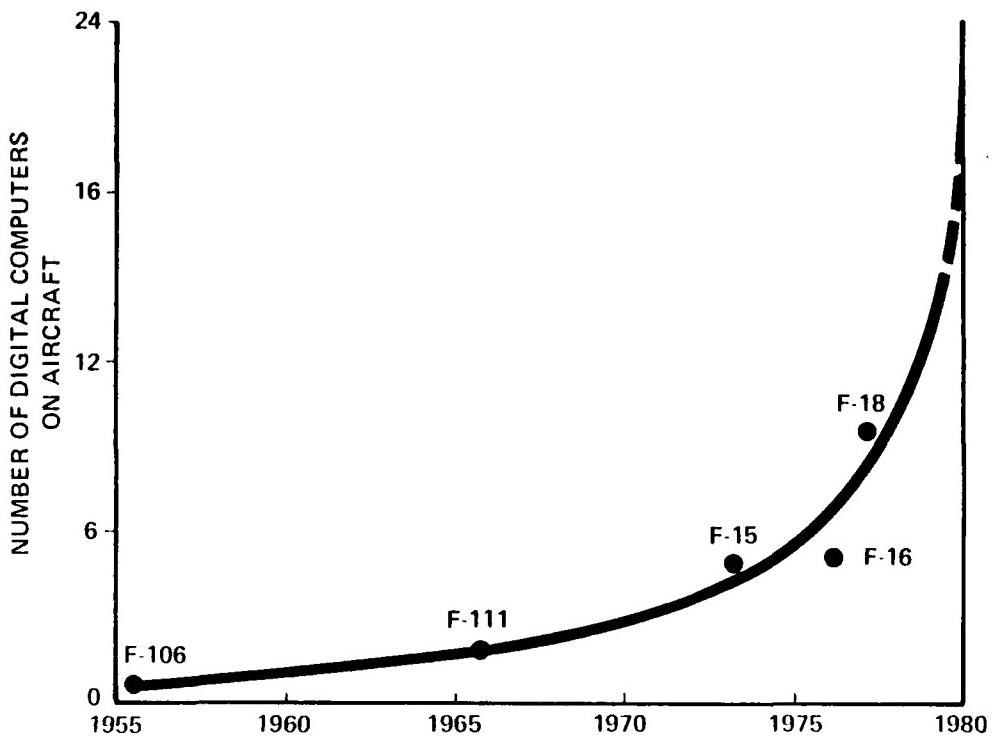
- Develop subsystems on chips using fabrication geometries of 1.25 microns (eventually .5 to .8 microns).
- Emphasis on increasing system throughput - ultimately real-time signal processing.
- Exploitation of high chip complexity, such as on-chip test, fault tolerance, error correction, etc.
- Achieve architectural concepts to minimize needs for custom designs.

VHSIC TECHNOLOGY CHOICES

<u>CIRCUIT TYPE</u>	<u>DENSITY</u>	<u>SPEEDS</u>	<u>PROBABILITY OF IMPROVEMENT</u>
ECL	LOW	HIGH	LOW
TTL	MODERATE	MODERATE	LOW
IIL	HIGH	MODERATE	Moderate
NMOS	HIGH	LOW	HIGH
CMOS	MODERATE	MODERATE	HIGH
GaAs	HIGH	HIGH	HIGH LONG TERM

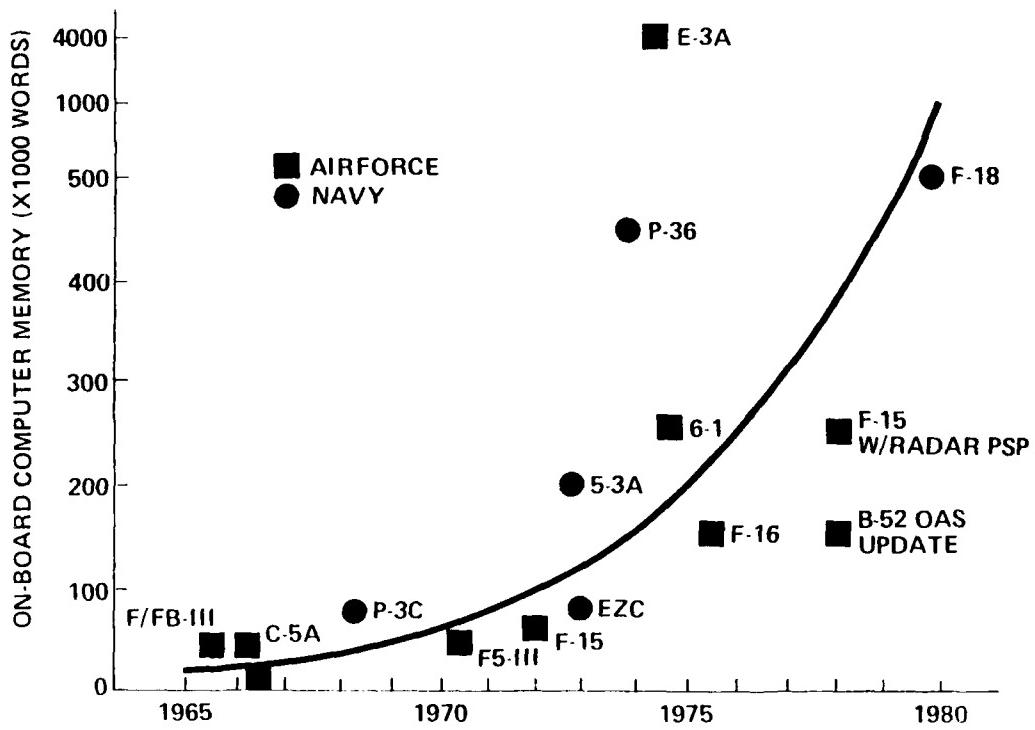
Shown here are the technology choices which are being evaluated by the VHSIC program. They include ECL, TTL, IIL, NMOS, CMOS, and GaAs. This chart shows the salient features of each in density, speed and probability for improvement.

TRENDS IN DIGITAL PROCESSING APPLICATIONS



This chart depicts just one type of military platform - aircraft - which typifies what is happening in embedded computers. Plotted here on a time scale is the number of on-board digital computers. The F-18 has no less than 12 digital computers.

GROWTH IN MILITARY AIRCRAFT SOFTWARE REQUIREMENTS



An indication of the software impact can be translated from the amount of on-board computer memory cells - shown here in thousands of computer memory words plotted on a time scale. The F-16 has about 150,000 memory words; the F-15 has over 200,000 words and the F-18 has nearly 500,000 words of on-board computer memory. And, of course, every memory word has data or software instructions associated with it.

"MAJOR NEW EMBEDDED SOFTWARE TECHNOLOGY INITIATIVE"

- **OBJECTIVE**
 - DAMPEN THE SOFTWARE COST SPIRAL
- **RATIONALE**
 - RISING SOFTWARE COSTS
 - NEW TOOLS NEEDED FOR SOFTWARE PROBLEMS
ANTICIPATED BY MID 80's
 - PECULIAR SOFTWARE NEEDS OF DEFENSE
 - NECESSARY TO REALIZE POTENTIAL BENEFITS OF ADA, i.e.:
 1. REDUCED DUPLICATION
 2. INTEROPERABILITY
- **TWO THRUSTS**
 - SHORT TERM – REALIZE BENEFITS OF ADA
 - LONG TERM – IMPROVE EFFECTIVENESS OF AUTOMATED SOFTWARE TECHNOLOGY
 - COMPLEMENT MID-80's HARDWARE

You have seen evidence in this presentation indicating that software is becoming very expensive indeed. To dampen the software cost spiral, we may see a "major new embedded software technology initiative."

Rationale behind such an initiative includes:

- rising software costs
- new tools are needed for software problems anticipated by the mid 80's
- defense has software needs different from industry
- necessary to realize the potential benefits of Ada, i.e., reduced duplication and interoperability.

This initiative could have two thrusts: in the near term - emphasis on realizing benefits of Ada. In the longer term - improved automated software effectiveness and to complement mid-80's hardware capabilities.

The following trends are foreseen:

Competition - We see greater pressure from Congress to increase competition in all embedded and ADP procurements.

Life Cycle Costs - Will receive more attention in the acquisition process.

Reduced Timetable - The weapons acquisition process is being looked at now to try to reduce the amount of time it takes to field a new weapon system.

Price/Performance - The trend toward reduced computer price/performance will continue to improve at approximately 16 percent per year and computer system reliability will improve at approximately 14 percent per year.

Training - Computer proliferation in the military inventory has created a training and logistics support problem.

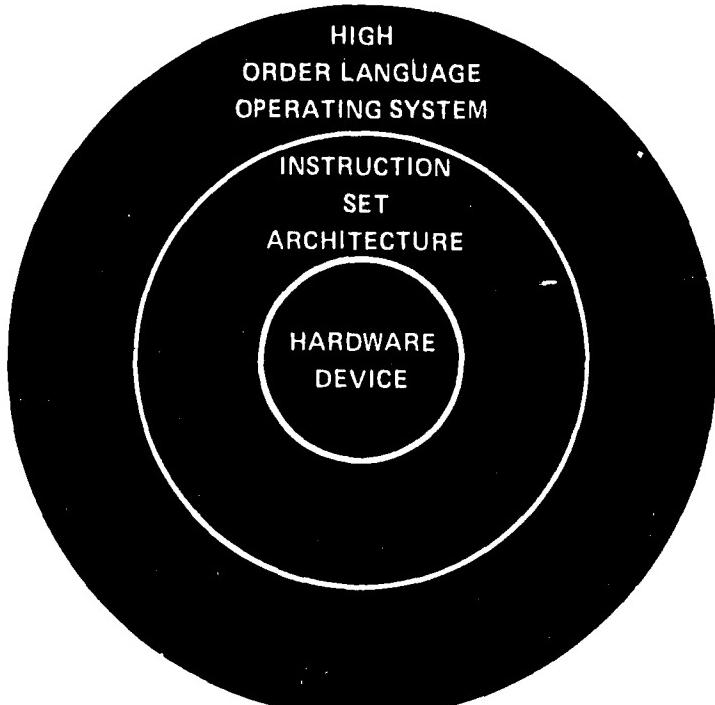
Software Personnel - The continuing proliferation of economic computer systems and the development of more sophisticated and complex systems will cause the demand for software personnel to become acute. At present, the escalating demand for computer systems personnel in the private industry and the all-volunteer force concept has begun to create manpower shortfalls in the military services. Faced with more attractive compensation by private industry, keeping qualified software personnel in the military is a serious problem.

In embedded computer hardware, there is a trend to move from standardizing at the "box" level to higher, non-hardware levels such as Instruction Set Architectures (ISA's) which could include accreditation/certification of hardware devices at a higher level.

Software is moving toward high level programming languages.

In logistics, the trend is toward hardware box level replacement.

LEVELS OF STANDARDIZATION



In conclusion, efforts to standardize in the embedded computer area have focused in the past on hardware devices such as the AN/UYK-20 or AN/AUK-14. The next level of standardization deals with the Instruction Set Architecture with hardware implementation/standardization of secondary importance. The trend, of course, is to standardize at higher levels such as the Ada HOL and associated operating system software. With the appropriate set of tools it is conceivable that the HOL level could be implemented with a variety of ISA's and hardware devices.

This concludes our "DOD Digital Data Processing" report.

This "DOD Digital Data Processing Study---a ten-year forecast" was performed under the auspices of the Requirements Committee, Government Division, Electronic Industries Association by an industry team. The results of the study was presented at the EIA Fall Symposium "The DOD Electronics Market---Forecast for the 80's" held in Los Angeles on October 7-9, 1980. David G. Stephan of Control Data Corporation chaired the study team, and Control Data Corporation has published and printed this copy of the study. Comments should be directed to Mr. Stephan at CDC, P.O. Box 0, Minneapolis, MN 55440, telephone (612) 853-5263.

Appendix B

Charter For The Post Development Software Support Activity

I. DESIGNATION OF ACTIVITY MANAGER

(Name of Individual) is designated as the (Name of Major Service Support Organization) Post Development Software Support Activity (PDSSA) Manager effective (Date). The PDSSA Manager reports to the Commanding General/Admiral (Service Command).

II. MISSION

The PDSSA Manager is responsible in accordance with Department of Defense (DoD) Directives (list as appropriate); Army, Navy, Air Force regulations (list as appropriate); and other pertinent regulations for:

- A. Providing software life cycle support, within the scope of this charter, for all assigned systems.
- B. Assessing and providing concurrence with the System Concept Paper (SCP)/Decision Coordinating Paper (DCP) and Acquisition Plan for Defense System Acquisition Review Council (DSARC) and (List corresponding service specific material acquisition decision process documentation) for adequacy of software life cycle support planning and executability.
- C. Supporting the System Acquisition Manager or his/her functional representative prior to transfer of responsibility for the operational life cycle support phases.

III. AUTHORITY AND RESPONSIBILITIES

A. The Activity Manager has been delegated the full line authority of the Commanding General/Admiral Service Command for the centralized management of the (Name of Major Service Support Organization) Post Development Software Support Activity.

B. Responsibilities

1. During the concept exploration phase, the Activity Manager is responsible for advanced software support planning, including system studies to assist/advise the acquisition manager or his functional representative in specifying broad bands of software supportability and support goals/requirements. Additional responsibility includes but is not limited to:

a. Identification and planning for compliance with existing Tactical Embedded Computer Resources (TECR) policy and standardization requirements pertaining to software supportability and support.

b. Analysis of Statement of Need and other available data for potential impact on software supportability and support (threat, mission, feasibility, risk, cost, trade-offs, etc.).

c. Determination of software logistic support requirements for inclusion in the system specification (or its equivalent).

d. Preparation of a draft Software Support Plan.

e. Coordination with the Integrated Logistic Support (ILS) function.

f. Software supportability and support requirements relative to currently defined interfaces between interfacing systems and subsystems.

g. Preliminary estimate of software support cost (including acquisition of any software support resources not otherwise projected to be available, and provision of software support over the projected operational life).

2. During the demonstration and validation phase, the Activity Manager is responsible for:

a. Completing and updating the Software Support Plan.

b. Coordination with the ILS function.

c. Performing software support studies to refine and define software support requirements, including security and software logistic support requirements in particular.

d. Determining software supportability requirements to be included in software performance specification (or equivalent). Examples include reliability, modularity, programming language/Ada compiler variant, etc.

e. Updating and refining software support cost estimates.

f. Determining the requirements (types, characteristics, numbers of and availability schedule) for PDSSA equipment, to include the following types:

(1) Computers (operational; trainer; ATE; compilation; integration and test; etc.).

(2) Simulators.

(3) Selected weapon system equipment items (e.g., sensors).

g. Coordination of assignment of PDSSA functions to

the PDSSA organizational, intermediate and depot maintenance levels; contractors; and other organizations (including inter-command and inter-service organizations).

h. Estimating PDSSA support personnel requirements (types, skill levels, numbers of each) for the following:

(1) Software Engineering (test, configuration management, quality assurance, requirements definition, design, etc.).

(2) Equipment Operators (computers, simulators, etc.).

(3) Maintenance (installation of replacement computer program units or modification to in-place units; failure verification; fault isolation; checkout of installed computer programs after replacement/modification; etc.)

i. Assuring that software supportability requirements are adequately defined and put in the contract, including the contract requirement for software supportability.

3. During the Full Scale Development Phase, the Activity Manager is responsible for:

a. Technical review of the system/subsystem contractor's engineering and development effort for continued software supportability.

b. Review the developing software and related hardware configuration items (CI's) to become prepared for assuming full post deployment support responsibility. As a minimum, this should include review of all software and related hardware technical data, safety requirements and the participation in reviews and audits. In particular, the reviews should include such design elements as: functional partitioning, coding, execute/operating system, structure, data base, intermodule communications design, etc. Additionally, the software production and maintenance facility requirements, and choices of programming languages and all related support software will be included, as well as the adequacy of the contractor's quality assurance system and configuration management procedures. These reviews and any appropriate recommendations will be coordinated with the cognizant contract administration office.

c. Provide requirements to the acquisition manager or his functional representative concerning necessary equipment facilities, support software, and other material necessary to place the PDSSA software/hardware facility in full operation. Provide budgetary information for all items recommended, and obtain assistance as required from the weapon system/subsystem contractor, software developer, and other contractors to provide details and supporting information.

d. Participate in software and related hardware engineering change impact analysis as appropriate, to ensure that proposed changes do not adversely affect supportability. The PDSSA will normally continue to perform this task throughout the life cycle of the system.

e. Participate in systems contractor software T&E program through the review of test plans and procedures, as well as acting as an observer during testing. The PDSSA may provide support to technical evaluation/operational evaluation test programs as requested, and upon completion of the development phase, will normally participate directly in the acceptance testing and audit of the software and related hardware CI product baselines. These tasks are performed as an agent of the acquisition manager or his functional representative.

f. Prepare or participate in the preparation of, the weapon system computer resource life cycle management plan (CRLCMP).

g. Plan for and, as specifically directed by the acquisition manager or his functional representative, initiate action to build up facilities, equipment, and manpower (suitably trained) to the extent necessary to assume full responsibility for the system computer/processor software and related hardware support program.

h. Plan for, arrange, and conduct appropriate training for PDSSA personnel. In order to provide the capability for the PDSSA to meet all system computer/processor software and related hardware operational and support problems, and adequately support the user, extensive training is required. For major systems, experience indicates that a training period of at least two to three years is necessary. Training should begin as early in the system full-scale development phase as feasible, and on-site location training of certain PDSSA personnel at the system contractor's facility will normally be required. The detailed requirements, plans, and schedules for PDSSA buildup and training must be included in the computer resource life cycle management plan (CRLMP) and other life cycle planning documents.

i. As directed by the acquisition manager or his functional representative, participate in computer/processor software and related hardware configuration management procedures in accordance with the CRLCMP. During the later stages of the system full-scale development phase, the computer/processor system software and related hardware may undergo frequent changes to correct deficiencies which become apparent during T&E. Proper configuration management is mandatory in order to ensure validity of tests and fully define the configuration of the software and hardware that are finally delivered to the user. While this phase of configuration management normally falls under the direction of the Design Agent (DA), the PDSSA may be required by the acquisition manager or his functional representative to closely monitor the contractor's configuration management procedures during this period to ensure effectiveness and also to become thoroughly familiar with the computer/processor software and related hardware configurations. During this period, the PDSSA will develop suitable configuration management procedures for in-house service use so that

they may be activated when the PDSSA assumes full software/related hardware support responsibilities. The PDSSA configuration management procedures must comply with the service requirements and will be scheduled for implementation in accordance with the plan indicated in the CRLCMP. It is important that the PDSSA monitor configuration management, and support the software configuration review board during the full-scale development phase, so that software configuration management can be properly transitioned in accordance with the CRLCMP.

j. Conduct appropriate review of software documentation contract deliverables as they become available to determine their quality, suitability, and acceptance based upon contract requirements and their true reflection of the software being delivered. The accuracy of the software documentation is extremely important as it becomes the baseline for use by the PDSSA, T&E activities, the service, and the user as well as for future software/hardware improvements and changes. The PDSSA will develop a detailed documentation management plan which will define procedures for receipt, verification, storage, duplication, distribution, inventory control, maintenance, and update.

k. Develop and prepare a detailed plan which will define procedures for assumption of responsibility for life cycle support of system computer/processor software and related hardware. This should include requirements and procedures for software inventory management, cross-indexing, storage, control, rapid retrieval, duplication, quality assurance, distribution, modification, and status accounting.

l. During the latter stages of the system full-scale development phase, a limited number of systems may be introduced to the user. The PDSSA will normally participate in user introduction at this time to prepare for assuming full responsibility in the computer/processor software and related hardware subsequent to deployment. During this time, the PDSSA will provide liaison with users for accomplishing submittal and analysis of system trouble reports. The PDSSA will distribute updated system software and associated documentation.

m. In preparation for assuming full support responsibility, the PDSSA may participate in software/hardware problem solving in support of the DA developer. The PDSSA may perform troubleshooting and may develop and test proposed solutions to the problem, providing such solutions to the DA developer as an alternative problem correction.

4. During the in-service support phase of the system life cycle, the PDSSA will:

a. Assume full responsibility for life cycle support of assigned system computer/processor software and related hardware. During the in-service phase of the system, the PDSSA fulfills the requirements of a software support activity. The PDSSA will be responsible for managing the computer/processor software and ensuring that changes conform to controlled specifications, and are verifiable

with other system functional areas and managers that might be impacted. The PDSSA will ensure that computer/processor software in-service engineering support is responsive to the needs of the user. The PDSSA will perform all of the following functions.

- (1) Rapid response to user software/hardware problems.
- (2) Problem tracking.
- (3) Problem analysis, including failure verification and fault isolation.
- (4) Problem resolution and impact analysis.
- (5) Development of corrections.
- (6) System enhancements through software changes.
- (7) Software configuration control
- (8) Verification, validation, functional integration testing, and performance assurance testing
- (9) Software production, distribution, and control
- (10) Determine where and how installation of changes will be accomplished
- (11) Software status accounting
- (12) User introduction training
- (13) Software documentation maintenance.

b. Be responsible for investigation of software/hardware problems and the initiation of corrective action. Prioritization of software problems and software trouble by degree of severity shall be performed. Approved software changes will be tested and verified prior to reproduction and distribution to receiving activities. These procedures will be in accordance with the information contained in the CRLCMP. Interface control documents are required to define relationships between the computer/processor system and other related systems. The PDSSA will review and recommend approval of all changes that affect these interface areas. The responsibility of the PDSSA extends to participation in problem solving at the interface level, and the testing of proposed solution that impacts the interface.

c. Assume responsibility for in-service engineering/logistics support of weapon system computer/processor software and related hardware.

d. Maintain and improve the software/hardware integration and test facility.

e. Provide continuing primary support to the acquisition manager or his functional representative and the user for assigned computer/processor software and related hardware as long as the system/subsystem remains in operation (until disposal).

IV. RESOURCE CONTROL

A. The Activity Manager will ensure that dollar and manpower

requirements to accomplish the above responsibilities are developed and submitted in accordance with established manpower/funding channels and procedures for inclusion in the Program Objective Memorandum (POM) for applicable target program years and that RDTE, procurement, operation and maintenance, and stock funds requirements are compatible at all times with the life cycle progression of assigned systems and provided in appropriate Work Breakdown Structure (WBS).

B. Monetary resources approved to accomplish the above responsibilities will be provided to the Activity Manager as direct mission funding for systems in the operational life cycle phase or by the participating organization having prime mission or task responsibility utilizing established funding channels and procedures. The Activity Manager will, in turn, provide the necessary funding, direction, or guidance, as applicable, to participating organizations for support provided in accordance with current regulations, policies, and procedures.

C. The Activity Manager will insure that the acquisition manager or his functional representative provides for two facilities early in the life cycle of the weapon system project: (1) A software production and maintenance facility; and (2) A software/hardware integration and test facility. These two facilities must be eventually located at, and operated by, the PDSSA.

D. PDSSA activities will ensure that the acquisition manager provides the facility with sufficient user equipment of all current versions being supported, to equip the software/hardware integration and test facility. The PDSSA facility will be considered as a field/fleet unit and will be assigned the highest Force Activity Designator justifiable under service guidelines.

V. LOCATION, SUPPORT AND STANDARDIZATION

A. Location and Support:

The (Major Service Support Organization) PDSSA is located at (Organization and Address) with necessary facilities and administrative support being provided by the organization. Liaison/field offices may be created by the Activity Manager within authorized funding as required without change of character.

B. Standardization:

The Activity Manager will:

1. Ensure that developing software systems will be designed with standardized interfaces for most efficient wartime software support and most cost effective use of established facilities and expertise.

2. Actively seek out and pursue opportunities for

promoting standardization and interoperability of assigned equipment(s) within PDSSA.

3. Incorporate interoperability requirements for all hardware and software to the maximum extent possible. (Pursue particularly electrical compatibility; mechanical interface; data and information transfer; and logistical supportability.)

4. As a minimum, review for applicability all relevant Standardization Agreements.

VI. COMMUNICATION CHANNELS

Direct communication is authorized among all participants involved in implementation of the development and support of assigned systems to ensure timely and effective direction and interchange of information among participants.

ANNEX A - LIST OF ASSIGNED SYSTEMS

(List of systems assigned to the Activity Manager)

Appendix C
Facilities Required For Post Deployment Software Support

C.1 GENERAL DESCRIPTION: A PDSSA facility is an engineering laboratory established for the evolutionary support of mission critical software.

The PDSS is made up of people, generic equipment, physical, environmental and communications facilities, data, documentation, and procedures. These facilities provide the capability to evolve software, simulate the operational environment, evaluate digital systems or subsystems, test software, integrate hardware and software, and address man/machine interfaces. PDSS facilities also provide the capability to maintain configuration "baselines" and manage activities and configuration items developed in the facility.

C.2 CAPABILITY AND FEATURES: the PDSS facility should have the following features and capabilities:

- 1) Support multiple mission critical systems within an integrated facility, including systems with multiple computers;
- 2) Support multiple functions with common modules;
- 3) Support harmonious interconnections of systems with dissimilar architectures, languages, program structures, and input/output requirements;
- 4) Support extension and reconfiguration, as the number of systems within the PDSS are increased or decreased;
- 5) Support mission critical software evolution through preplanned product improvements;
- 6) Support life cycle management and systems engineering cost objectives;
- 7) Incorporate existing support assets into the PDSS facility;
- 8) Meet physical, technical, and administrative security requirements.
- 9) Support rapid response to mission critical reprogramming requirements.

The PDSS facility must be capable of simulating the operational environment as nearly as possible. In order to verify the interfaces with other systems, processors, or hardware components, the PDSS facility must these devices or be capable of being configured to access them at other PDSS sites or remote locations.

Exercising a system in the real world environment is the only true test of its effectiveness. When navigation, detection and tracking capabilities are exercised such that a target is detected, a weapon is launched and the target is no longer a threat - this is the most realistic measure of a weapons system effectiveness. This, however, is not possible in most cases since political, economic, environmental, humanistic, or operational constraints usually prohibit real world exercises. Simulation has proven to be the next best avenue. It is used during development and testing of the original system and is also used to train and exercise operators. The simulation used in such training devices have become an essential requirement for successful system deployment.

The same real world requirements placed on those mission critical system trainers must be extended to their respective PDSS. In fact, the integration of a system trainer and its PDSS could prove economically and programmatically beneficial.

C.3 SUPPORT TOOLS: To maximize the productivity of personnel, enhance the quality of delivered systems and improve responsiveness to changing requirements, PDSS facilities should have a set of standard automated tools with predefined interfaces to allow inter-tool communication. These tools should be capable of supporting specific PDSS processes in concert with generic support processes. Automated PDSS tools should support the following functions:

- 1) Data Base Management.
- 2) Configuration Management.
- 3) Information Management.
- 4) Requirements Analysis.
- 5) Specifications Definition.
- 6) Software Design.

- 7) Documentation.
- 8) Software Development.
- 9) Software Test.
- 10) System Integration.
- 11) Simulation.
- 12) Validation and Verification.
- 13) Project and Cost Management.

These tools must be provided in concert with high order languages preferably Ada and their automated support environments for the management, development, reengineering, test and integration of mission critical software.

C.4 PHYSICAL FACILITY: To prevent premature technical obsolescence, the PDSS facility should have an open core design, raised floors, relocatable walls, floor to floor chases, environmental and electrical distribution systems. Building flexibility into the initial design will allow the PDSS facility to be reconfigured to meet future tasks. The electrical system must meet the requirements of commercial computer systems, avionics and unique weapon systems and where security is a requirement, all power must be filtered. Various special requirements must be met based on specific applications, such as requiring geographical benchmarks, special fire protection, special grounding or security needs. General PDSS facility requirements include:

- 1) Office space for engineers, computer scientists and support personnel.
- 2) Local area and long haul network nodes.
- 3) Power and cooling equipment for mission critical hardware systems and PDSS support systems.
- 4) Vaults for security and physical protection.
- 5) Libraries for magnetic and printed media.
- 6) Equipment maintenance areas.
- 7) Growth/storage space, and
- 8) Security planning.

C.5 SECURITY PLANNING REQUIREMENTS: Supporting mission critical software has major security related cost impacts. Although the classification requirements can vary from unclassified to compartmented sensitive, only a thorough risk analysis can determine the extent of physical, technical, and administrative security measures which should be employed within the PDSS. Security protections must be considered for systems that contain or process classified information or interface with other systems that do. PDSS facilities must be designed to accommodate the current security requirements of the system and be upgradable to provide higher levels of protection to satisfy future requirements of the system and its interfaces. The following security considerations must be included in all PDSS FACILITIES:

- 1) Security accreditation requirements of cognizant authorities to operate PDSS facility.
- 2) Use of cryptographic systems for data transmission outside the facility.
- 3) Administrative controls to select, evaluate and monitor the personnel to whom access will be granted.
- 4) Physical controls as determined by the risk analysis.
- 5) Technical controls for all computers, displays and peripheral equipment such as TEMPEST, multi-level security, software access controls, etc.

Appendix D
List of Panel Members Arranged By Subpanel

PANEL C - COST OF OWNERSHIP

CO-CHAIRMEN: Riley, LTC J. (Jim) USAF
HO AFSC/DIA, Andrews AFB
Washington, DC 20334
(301) 981-2482
A/V 858-2482/7164

Sievert, Mr. G. (Gene)
Teledyne-Brown Engineering
300 Sparkman Dr.
Huntsville, AL 35807
(205) 532-1500

SUBPANEL 1: CURRENT SERVICE APPROACHES

CHAIRMAN Pollard, LCOL R. L. (Ray) USMC
Requirements & Programs Br.
Developmental Coordination Div.
Development Center, MCDEC
Quantico, VA 22134
(703) 640-2873
A/V 278-2873

MEMBERS: Goldsmith, Mr. L. (Len)
AMCCOM/DRSMC-TSB(D)
Dover, NJ 07801
(202) 328-3320
A/V 880-3320

Stewart, CDR J. C. (John) USN
Naval Sea Systems Command
(PMS-408)
Washington, DC 20360
(202) 692-8204
A/V 222-8204

Schmidt, Mr. J. (Jerry)
LOC/CF
Wright Patterson AFB, OH 45433
(513) 257-6637
A/V 787-6637

SUBPANEL 2: PDSS CHARTER

CHAIRMAN Branyan, Mr. E. (Elmer)
General Electric Company
Military Programs Dept.
P.O. Box 8048
Philadelphia, PA 19101
(215) 962-4735

MEMBERS: Hofer, Dr. R. (Ron)
PM Trade, DRCPM-TND
Naval Training Equipment Center
Orlando, FL 32813
(305) 646-5779
A/V 791-5779

Bremhorst, CDR J. (Joe) USN
Naval Air Systems Command
(AIR-5432)
Washington, DC 20360
(202) 746-0650
A/V 286-0605

SUBPANEL 2: PDSS CHARTER (Continued)

Woods, Mr. D. (Dennis)
Software Enterprises Corp.
31220 LaBaya Drive, Suite 110
Westlake Village, CA 91362
(213) 889-7814

Gordon, Mr. C. (Chuck)
CACI Inc., Federal Penthouse
1700 N. Moore St.
Arlington, VA 22209
(703) 276-2838

SUBPANEL 3: FACILITIES REQUIRED

CHAIRMAN: Steele, Mr. R. (Russell)
TRW
P.O. Box 1058
North Highland, CA 95660
(916) 920-2613

MEMBERS: Forsythe, Mr. R. (Roger)
Naval Electronic Sys. Command
(PME-120-3)
Washington, DC 20363
(202) 433-4581
A/V 288-4581

Simpson, CDR R. J. (Dick) USN
Chief of Naval Operations
(OP-945D)
Pentagon, RM. 5E572
Washington, DC 20350
(202) 697-6494
A/V 227-6494

SUBPANEL 4: COST SAVING RECOMMENDATIONS

CHAIRMAN: Smith, Mr. W. (Bill)
OASN (RE&S), Pentagon, 5E785
Washington, DC 20350
(202) 694-4691
A/V 224-4691

MEMBERS: Renfro, COL R. (Ron) USA
U.S. Army Artillery School &
Center (DRCPM-TF-FS)
Ft. Sill, OK 73503
(405) 351-4200
A/V 639-6850

MacDonald, Mr. B. (Bobby)
WR-ALC/MMRR
Robins AFB, GA 31098
(912) 926-4525
A/V 468-4525

Solomond, Dr. J. (John)
HQ, DARCOM/DRCDE-SB
5001 Eisenhower Avenue
Alexandria, VA 22333
(202) 274-9318
A/V 284-9318

Mellin, Mr. J. P. (Pat)
Control Data Corporation
(M/S HQNIDT)
3101 East 80th St.
P.O. Box 609
Minneapolis, MN 55440
(612) 853-6639

ORLANDO I

FINAL REPORT

PANEL D

POST DEVELOPMENT SOFTWARE SUPPORT (PDSS)

SOFTWARE SUPPORT ENVIRONMENT

Co-Chairman: Mr. Jim Hess
HQ DARCOM/DRCDE-SB
5001 Eisenhower Avenue
Alexandria, VA 22333
(202) 274-9318
A/V 284-9318

Co-Chairman: Mr. Jerry Raveling
Sperry Corporation
Computer Systems, M.S. U1E13
P.O. Box 43525
St. Paul, MN 55164
(612) 456-3545

RD-R199 109 JOINT LOGISTICS COMMANDERS' WORKSHOP ON POST-DEPLOYMENT 4/6
SOFTWARE SUPPORT (U) JOINT POLICY COORDINATING GROUP
ON COMPUTER RESOURCE MANAGEMENT JUN 84

UNCLASSIFIED

F/G 12/3

NL

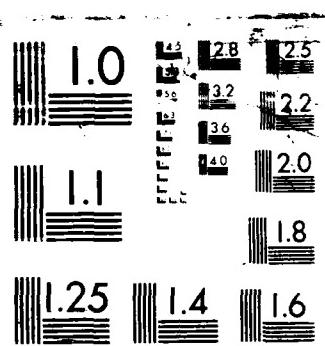


TABLE OF CONTENTS

Report of the Panel on Software Support Environment

	<u>Page</u>
Table of Contents	
List of Figures	
List of Tables	
Appendices	
4.4 SOFTWARE SUPPORT ENVIRONMENT	4-4-1
4.4.1 OBJECTIVES	4-4-2
4.4.2 SCOPE	4-4-3
4.4.3 APPROACH	4-4-3
4.4.3.1 Preparation	4-4-3
4.4.3.2 Panel Organization and Operation	4-4-4
4.4.3.3 Issues	4-4-4
4.4.4 DISCUSSION	4-4-4
4.4.4.1 Subpanel 1 - Definition of the Support Software Environment	4-4-4
4.4.4.1.1 Introduction	4-4-4
4.4.4.1.2 The PDSS Environment	4-4-5
4.4.4.1.2.1 Description and Characterization of the Environment	4-4-5
4.4.4.1.2.2 Environment Architectural View	4-4-5
4.4.4.1.2.3 Life Cycle View	4-4-9
4.4.4.1.2.4 Tool Set Viewpoint	4-4-9
4.4.4.1.2.5 Other Perspectives	4-4-12
4.4.4.1.3 Definition of PDSS Baseline Requirements	4-4-12
4.4.4.1.4 Discussion	4-4-14
4.4.4.2 Subpanel 2 - Management Support Systems	4-4-19
4.4.4.2.1 Management Support Systems	4-4-20
4.4.4.2.2 Security Implications/Requirements	4-4-21
4.4.4.2.3 Provisions of the Support Environment	4-4-22
4.4.4.3 Subpanel 3 - Contractual Provisions	4-4-23
4.4.4.3.1 Project Data Base	4-4-24
4.4.4.3.2 Testing Processes Development and History	4-4-25
4.4.4.3.3 PDSS Environment Acquisition	4-4-25
4.4.4.3.4 PDSS Reference Manual	4-4-28
4.4.4.3.5 Tools Which are Unique to the PDSS	4-4-28
4.4.4.4 Subpanel 4 - Application Area Unique Criteria	4-4-33
4.4.4.4.1 SSE Requirements	4-4-33
4.4.4.4.2 Operational Environment Considerations	4-4-34
4.4.4.4.3 Impact of Generic SSE on Resources	4-4-34
4.4.4.5 Conclusions and Recommendations	4-4-35
4.4.4.5.1 Subpanel 1 - Definition of the Support Software Environment	4-4-35
4.4.4.5.1.1 Environmental Architecture View	4-4-35
4.4.4.5.1.2 Life Cycle View	4-4-35
4.4.4.5.1.3 Other Perspectives	4-4-35
4.4.4.5.1.4 Definition of PDSS Baseline Requirements	4-4-36

TABLE OF CONTENTS (continued)

	<u>Page</u>	
4.4.5.2	Subpanel 2 - Management Support Systems	4-4-36
4.4.5.2.1	Security Requirements	4-4-36
4.4.5.3	Subpanel 3 - Contractual Provisions	4-4-36
4.4.5.3.1	Project Data Base	4-4-36
4.4.5.3.2	PDSS Environment Acquisition	4-4-40
4.4.5.3.3	PDSS Reference Manual	4-4-41
4.4.5.3.4	Tools Which Are Unique to the PDSS	4-4-41
4.4.5.4	Subpanel 4 - Application Area Unique Criteria Recommendations	4-4-41

LIST OF FIGURES

	<u>Page</u>
Figure 4.4-1 STARS View of the Software Environment	4-4-6
4.4-2 Post Deployment Software Support Environment	4-4-7
4.4-3 The Environment	4-4-8
4.4-4 Life Cycle View of the SEE	4-4-10
4.4-5 Tool Set View of the Software Engineering Environment	4-4-11
4.4-6 Accommodating Non-Generic Tools in a Generic Framework via MIL-STD-SDS	4-4-13
4.4-7 PDSS Facility	4-4-15
4.4-8 PDSS Facility Capabilities	4-4-16
4.4-9 Ideal PDDS Host Computer System	4-4-18

LIST OF TABLES

Table 4.4-1	PDSS Manual Outline	4-4-29
4.4-2	PDSS/SEATECS Mapping	4-4-37

APPENDICES

Appendix A	Panel D Participants	4-4-A-1
B	Bibliography	4-4-B-1
C	Panel Presentation Summaries	4-4-C-1
D	A Software Engineering Environment (SEE) for Weapons System Software	4-4-D-1

4.4 SOFTWARE SUPPORT ENVIRONMENT

Over the past decade there has been a dramatic increase in the number of planned and deployed Mission Critical Computer Systems (MCCS). A MCCS is a system which is of significant importance and which is integral to the effectiveness of today's military combat and support systems. They include airborne, fixed and mobile ground, surface and sub-surface naval, and space systems which are required to operate in both hostile and benign environments. MCCS's are generally characterized as ruggedized programmable devices which exhibit high speed, accuracy, and reliability in the processing and manipulation of data, performance of computations, and in the exercising of system control. These features have, and will continue to contribute to the development of military systems which meet or exceed performance, reliability, and maintainability requirements, and which demonstrate flexibility when responding to new requirements. MCCS's implement or aid-in the implementation of system and subsystem performance characteristics, and serve to integrate the various system elements into highly responsive and effective systems. MCCS's, through their programmability features, provide military systems with improved flexibility to respond to changing operational requirements. The embedded computer, in most instances, executes software. Thus, MCCS functions can be easily modified and/or enhanced by modifying (or replacing) the software. Normally software can be modified much faster and at a fraction of the cost of that which would be required to implement a comparable change in hardware.

With the continued improvement in the cost/performance ratio for computer hardware, and improvements in computer software capabilities, the military services are able to develop and deploy more-and-more complex systems. At the same time, this dramatic expansion in the use of MCCS is creating new and continually expanding logistic support requirements. All of the Services are confronted with the problem of supporting a rapidly expanding number of unique computer based systems. Each unique MCSS, brings with it its own Instruction Set Architecture (ISA), hardware spare parts requirements, and related support and applications software. Further, because of the inherent complexity and interdependence of current classes of MCCS, support requirements extend beyond the MCCS itself to encompass supporting or interfacing systems; for example, the Automatic Test Equipment (ATE) and the training simulator which support/interface with a combat weapons system.

The logistics support problem for MCCS has been further exacerbated in recent years through the introduction of microprocessor based embedded subsystems/systems. These systems are also normally reprogrammable. Inherent to the microprocessor-based applications, however, is the increasing tendency to substitute hardware functionality for software functionality. The growing use of firmware requires additional development, support, and test environments above and beyond the software environmental requirements.

4.4.1 OBJECTIVES

MCCS software serves to modify, enhance, and integrate the processing system into a functional system, the MCCS software controls the capability of the system. Thus, the MCCS relates directly to the system's mission availability and accomplishment. Today's military, in most instances, cannot perform their mission without full reliance upon the MCCS software which is inherent to their operational systems.

To effectively and efficiently modify MCCS software and in-general provide engineering support for the MCCS requires specialized facilities, skills, and equipment. After the acquisition of the operational (or test, training, etc.) systems has been completed and the system has been deployed to its operational environment, the Military Services commands/organizations assume responsibility for post development engineering and support. Often this (post development phase of the MCCS life cycle will encompass a period of from ten (10) to twenty (20) years, or more.

Each of the Services, the commands within the Services, and other government agencies have established and operate unique MCCS support facilities. These facilities have been designed and configured to meet Service, command, or system unique requirements. The facilities range in size and sophistication dependent on the support requirements of each MCCS and the number of separate MCCS's to be supported at each facility.

The principal difference in post development Software Support Environment (SSE's) is related to the basic maintenance concept established for a system and its major subsystems; i.e., will support be centralized or decentralized and what level of system (or subsystem) support will be provided. Within each basic maintenance concept category, there are significant variances in approach; e.g., the system or subsystem may be managed as individual entities or in some combination, by equipment function, by tactical function, etc. Other unique differences in support environments can usually be traced to the basic maintenance concept adopted by the Services.

With the development of Ada¹, and the emphasis placed on tools by the current DOD STARS² program there has been increased interest expressed in establishing common SSE's. The assumption is that a reduction in the number of Service, Command and project unique environments would lead to greater cost efficiencies and improved productivity.

The panel's basic objectives were to define the requirements for a generic PDSS software support environment (SSE), and to assess the commonality of requirements with DOD-sponsored, development-oriented software environments.

¹ Ada is a trademark of the U.S. Department of Defense.

² Software technology for Adaptable, Reliable Systems.

4.4.2 SCOPE

The panel was assigned to discuss selected aspects of a generic PDSS environment. These aspects were addressed to the panel in the form of a series of questions which dealt with:

- o Requirements for defining a core SSE generic equipment/software suite.
- o Management support systems requirements to include criteria for GFE/CFE, security, and PDSS versus development environments.
- o Major contractual considerations which must be addressed in the system acquisition and post development phases of the life cycle.
- o Whether the type of software to be supported by the PDSS facility places unique requirements on the SSE.

4.4.3 APPROACH

4.4.3.1 Preparation

The key to effective workshops lies first and foremost in the selection of panel participants. The success of Panel D was assured by the quality of the personnel selected for the panel. The individuals identified in Appendix A of this report represented some of the best that the government and industry could provide to discuss SSE issues. Prior to the workshop each member of the panel was solicited for suggestions on the panel charter, planned approach, and to provide reference materials which would promote/support the panel discussions. Through this methodology, and through the independent work of the Co-Chairs a library of contemporary SSE material was assembled for use by the panel. See Appendix B, Bibliography for a list of this data.

Early in the panel planning process it was determined that the work of the panel would be enhanced if an overview of current DOD/Services PDSS environments was presented. The panel Co-Chairs made arrangements for five briefings to be presented on the first full day of the workshop. Briefings presented were:

- o "A Builders Guide to Software Engineering Environments", Mr. William E. (Bill) Riddle, Software Design and Analysis, Boulder, Co.
- o "A Modern Facility for Software Production and Maintenance", Mr. H.G. (Hank) Stuebing, Naval Air Development Center, Warminster, Pa.
- o "Electronic Warfare Avionics Integration Support Facility", Mr. J.J. (John) La Vecchia, AFLC Robins AFB, Ga.
- o "United States Army, Post Deployment Software Support (PDSS) Study", J. (Jim) Hess, DARCOM, Alexandria, Va.
- o "A Software Engineering Environment (SEE) for Weapon System Software", H.G. (Hank) Stuebing, Naval Air Development Center, Warminster, PA.

See Appendix C, Panel Presentation Summaries for a synopsis of each of the five briefings.

4.4.3.2 Panel Organization, Topics and Operation

At the conclusion of the above briefings, the Panel broke into four sub-panels. Subpanel topics were refined (based on the questions listed in Section 4.4.2 SCOPE) and Subpanel Leaders and Recorders were selected by the subpanel members:

Subpanel 1: Barry Boehm (TRW)--Co-chair
George Sumrall (US Army)--Co-chair/Recorder
Topic: Definition of Support Software Environment (SSE)
Subpanel 2: John Cole (US Army)--Chair
Bob Sauer (USMC)--Recorder
Topic: Management Support Systems Considerations
Subpanel 3: Skip Meiers (USCG)--Chair/Recorder
Topic: Contractual Provisions
Subpanel 4: John Martinsen (Boeing)--Chair/Recorder
Topic: Application Area Unique Criteria

Subpanels discussed their assigned topic areas, prepared written notes on major items of discussion, and developed subpanel conclusions/recommendations.

The subpanels reformed for full panel sessions late in the morning on the second day (Wednesday) and again on the afternoon of the fourth day (Thursday). Subpanel Leaders provided a brief review of the subpanels deliberations, and reported on the preliminary recommendations developed by the subpanel.

The Co-Chairs prepared, based on these subpanel reports, an SSE Panel summary for presentation at the full workshop joint sessions on Tuesday and Wednesday afternoons, and on Friday morning. These briefings summarized the panel's work, provided a review of the panel's conclusions/recommendations, and summarized other germane/salient information.

4.4.3.3 Issues

A summary of the four Panel D subpanels is presented in the following paragraphs. The subpanel summaries address the basic issues identified in the Panel Charter as described in paragraphs 4.4.1 and 4.4.2 above, and provide a synopsis of the subpanels discussions. Major subpanel/panel conclusions and recommendations are presented in the final (paragraph 4.4.5) section of this report.

4.4.4 DISCUSSION

4.4.4.1 SUBPANEL 1 - Definition of Support Software Environment (SSE)

4.4.4.1.1 Introduction

The subpanel's objective was to define the requirements for establishing an effective generic PDSS environment. Early on, the subpanel determined that

the PDSS environment should be addressed in the large framework (super-structure) of the total system life cycle and its environments (see Fig 4.4-1). In the process of analyzing PDSS requirements within this super structure, the subpanel determined three key points: 1) there are a set of generic functions that are common across the environment; 2) that requirements specification and development engineering (H/W S/W) functions also occur (albeit with a different emphasis) in the post development cycle; and 3) "products" produced in upstream engineering cycles were critical to effective post-deployment support. Thus, the subpanel believes that the JLC/CRM should strive for a generic engineering environment that supports not only the PDSS but the entire system life cycle. This "goal system", however, should be developed in a block evolution building on work which is already underway (see Section 4.4.4.1.3) and on initial requirements that are clearly understood.

4.4.4.1.2 The PDSS Environment

The term "environment" in the PDSS context refers to an integrated, coherent collection of tools (mostly software) which support activities encountered in the post deployment software support process. It is the opinion of the PDSS environment panel that this environment is not substantially different from the software engineering environment (SEE) being addressed by the STARS program*. In fact, if the PDSS needs are addressed in the STARS SEE definition, then that environment would provide a consistency of operation across the life cycle.

4.4.4.1.2.1 Description and Characterization of the Environment

To understand the characteristics of the software support environment (SSE) it is useful to view the SSE from different perspectives. Fig 4.4-2 suggests three different view points for discussion/understanding the environment:

- o The environment architectural view
- o The life cycle view
- o The environment tool set view

4.4.4.1.2.2 Environment Architectural View

The SSE as seen from the architectural viewpoint is shown in Figure 4.4-3. This view presents to the various classes of SSE users (system engineers, programmers, system testers, managers, etc.,), the overall layout of the SSE functions and a concept of how these functions would be used.

As indicated in Figure 4.4-3, the SSE includes not only software tools and entities, but also the communications interface equipment and protocols necessary to link the software and host-machine capabilities to the various target machines, target environments, test drivers, and instrumentation capabilities. In this form the SSE supports the software/system integration and test functions required for PDSS as well as the standard software development and modification functions.

* See Appendix C, briefing number 5.

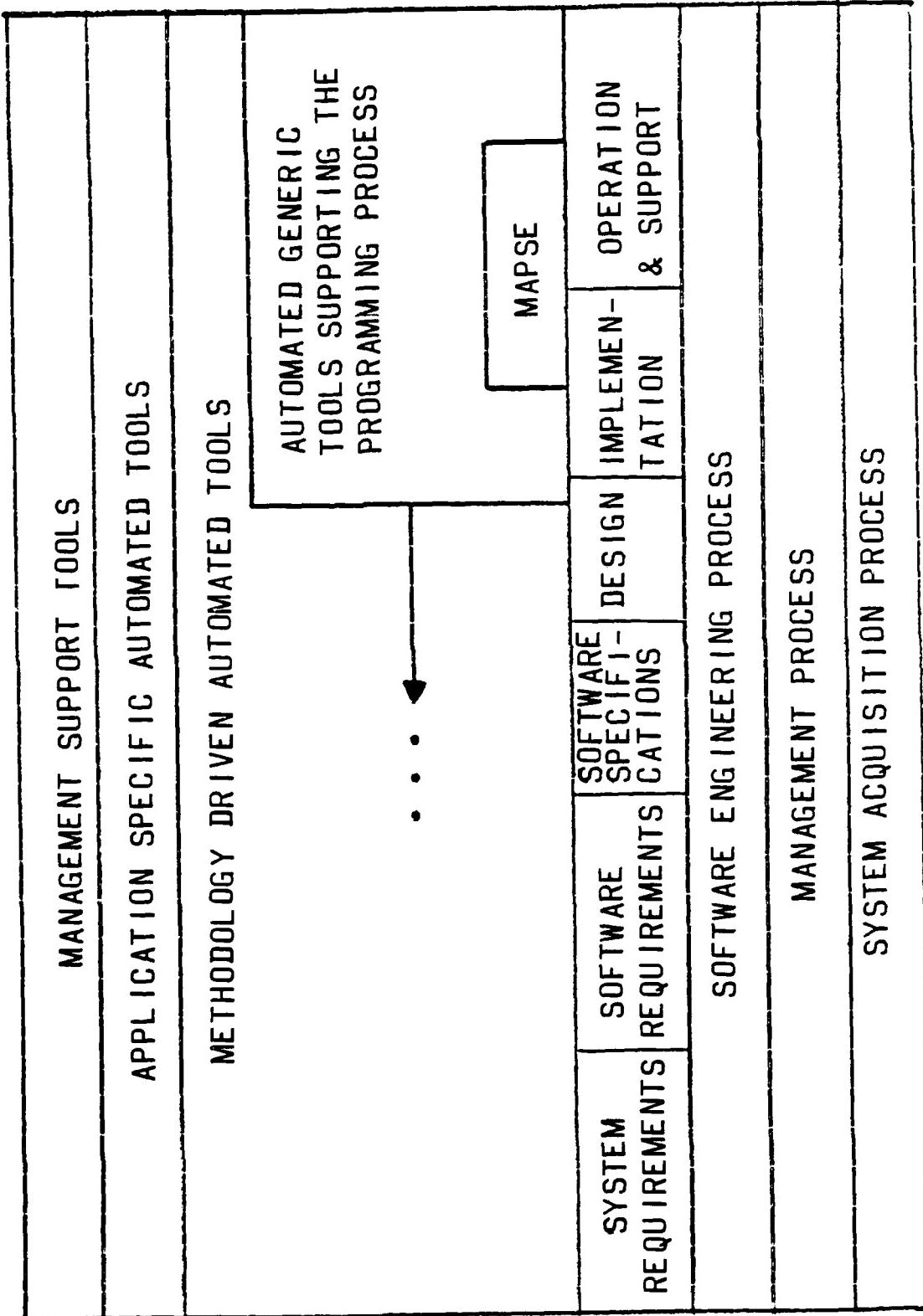


Figure 4.4-1 Stars View of the Software Environment

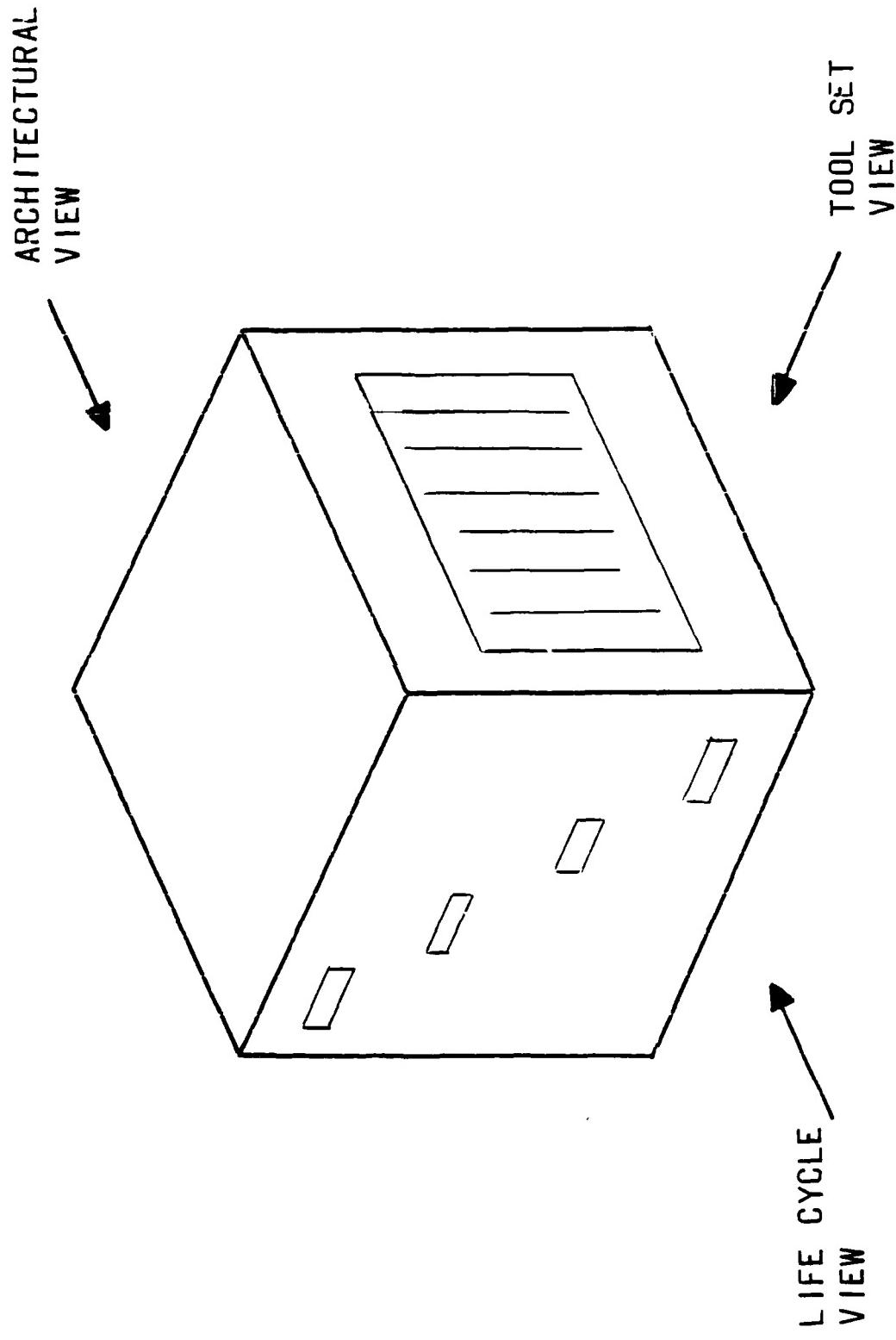


Figure 4.4-2 Post Deployment Software Support Environment

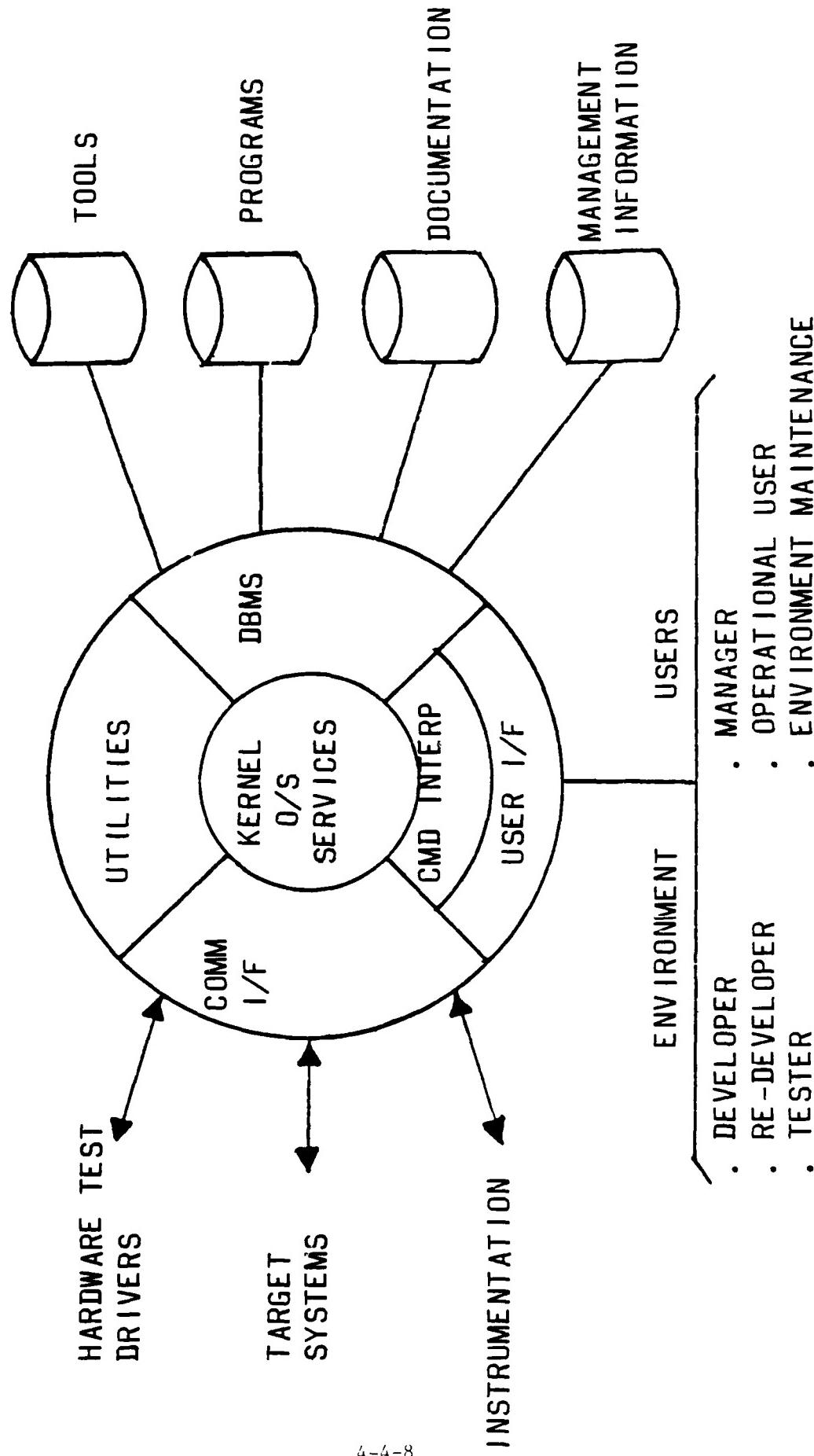


Figure 4.4-3 The Environment

The overall concept of operation indicated in Fig 4.4-3 involves the ability of each user or class of users to develop a tailored, personalized mix of basic SSE commands and macro-commands which best help perform their PDSS functions. Thus, a programmer's commands would focus on retrieving program or design entities, and invoking combinations of editors, formatters, static analyzers, compilers, linkers, loaders, and execution monitors which help him develop, test, and document his program. On the other hand, a system test engineer's commands would focus on configuring target computers and associated hardware or simulated equipment with test-driver and instrumentation equipment, retrieving and loading programs and data, and invoking a test scenario involving the exercise of various system capabilities, the monitoring and analysis of test results, and the presentation of test results in desired reporting formats.

In this concept of operation, the command interpreter in Fig 4.4-3 operates on the user's commands to produce a sequence of internal commands to the SSE's DBMS, utilities, and communications interface to perform the functions desired by the user.

4.4.4.1.2.3 Life Cycle View

Another useful view of the SSE is the life cycle as defined in MIL-STD-SDS and shown in Fig 4.4-4. This view indicates that the environment consists of functions and capabilities which support each of the life cycle phases, and a set of functions and capabilities which span the entire life-cycle.

One issue which can be addressed from this view is the extent of commonality between a development SSE and a post development SSE. As indicated by the cross-hatching and brackets in Fig 4.4-4, the commonalities vastly outweigh the differences.

4.4.4.1.2.4 Tool Set View Point

As stated earlier, the SSE consists of a consistent, coherent collection of tools, structured so as to promote communication between tools from a data and control standpoint. Fig 4.4-5 shows the SSE as seen from the tool viewpoint. Tool sets are grouped into a number of layers. The core set of tools consist of the more generic basic tools which support the host computer operation, including the routine system, system administration, user interaction data management, access rights, security, etc.

The core also supports and allows additional tool sets to be "expanded" in providing for a multiple language capability, methodology dependent tools, tools specific to specific applications (e.g., EW, avionics) and management support tools. These higher level layers provide the capability to accomodate multiple languages, multiple methodologies and to "tailor" the environment to particular applications.

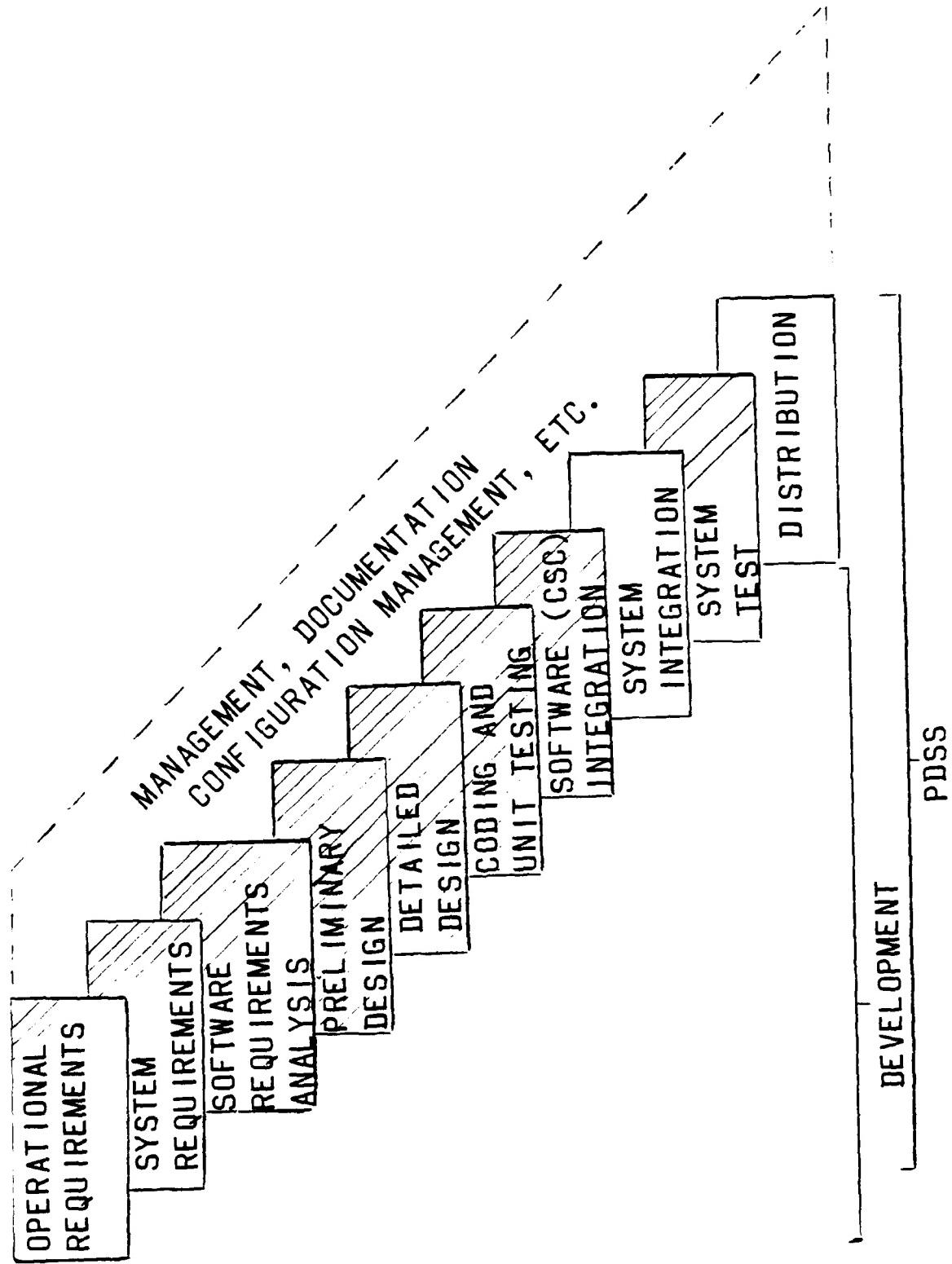


Figure 4.4-4 Life Cycle View of the SEE

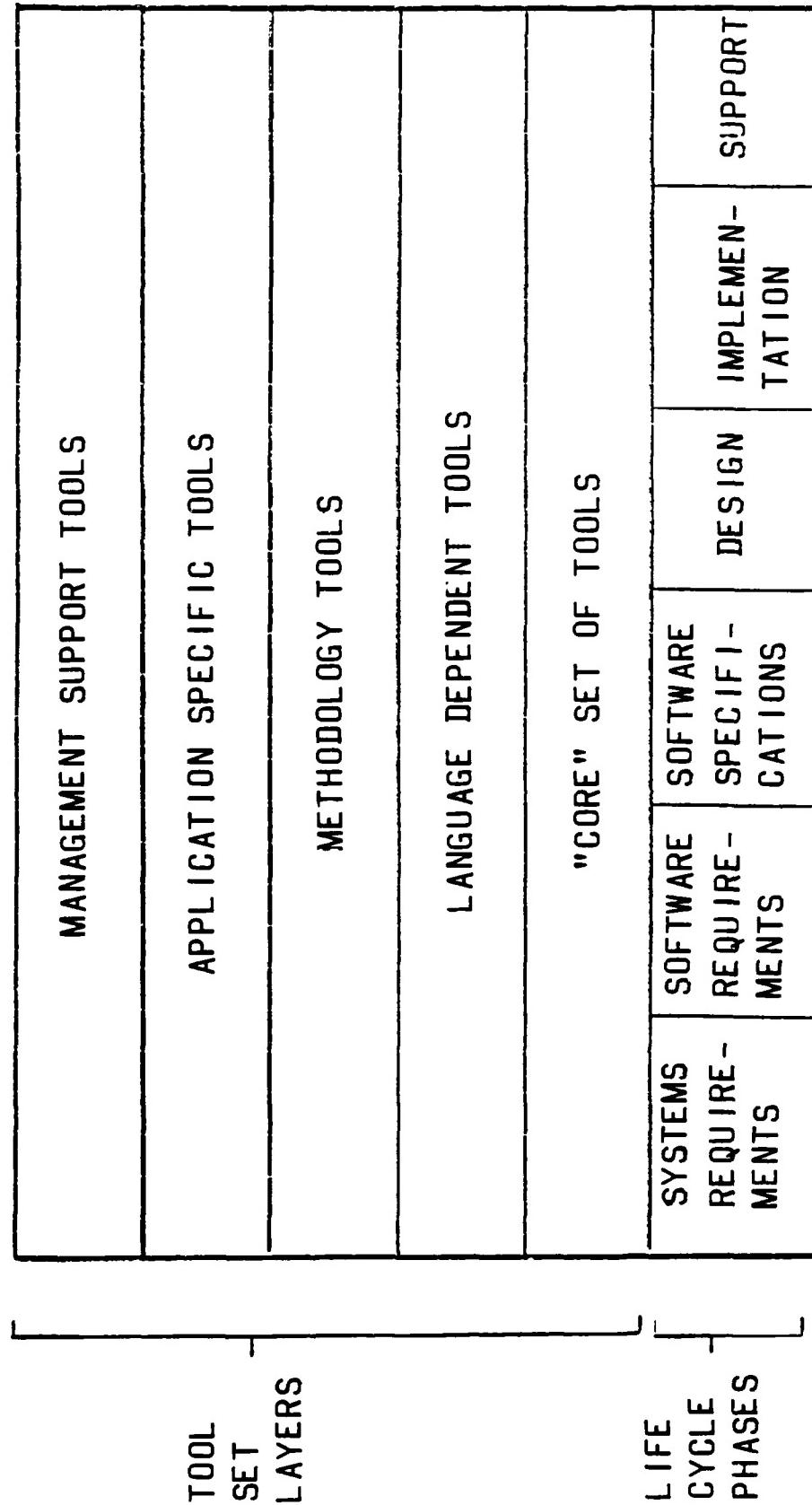


Figure 4.4-5 Tool Set View of the Software Engineering Environment

4.4.4.1.2.5 Other Perspectives

Besides the SSE views shown in the preceding figures, there are some other views which provide useful perspectives. One is the DOD-level ownership views shown in Figure 4.4.1, which expands on the layered view of the SSE shown in Fig 4.4-5. The view in Fig 4.4-6 shows that, across the DOD, there will be SSEs which contain different combinations of:

- o Several different requirements specification languages (RLs): PSL, RSL, Parras A-7, etc.
- o Several different design languages (DLs): PDLs, Ada PDLs, structure charts, HIPOs, etc.
- o Several different programming languages (PLs): Ada, JOVIAL, FORTAN, CMS-2, etc.
- o Different Integration and Test (I&T) tools.
- o Several types of management support systems (C/SCSC, C/SSR, OSCAR, PERT/COST, etc.).

From a DOD-level ownership view, it will be extremely important to facilitate the interoperability and commonality of SSEs containing different combinations of the above languages and tools. It is valuable to consider the columns in Fig 4.4-6 as sources of variation whose details (following the principles of information hiding) should be hidden as much as possible from each other. A good start in this direction is provided by MIL-STD-SDS. For example:

- o Variations between details in the content of requirements language are hidden by the MIL-STD-SDS provision that each itemized requirement have a unique identifier. Thus, one can perform requirements traceability functions in a manner independent of the detailed content of each itemized requirement.
- o Variations between details in the content of design languages are similarly hidden by the MIL-STD-SDS provisions on the identification of individual Computer Software Components (CSCs) and units.

Similarly, a good set of interface definitions and conventions can promote:

- o Host-target interoperability via standard network-interface protocols.
- o Management support interoperability via standard WBS element definitions, milestone definitions, etc.
- o Common support of data base, documentation, and CM functions via standard data base object definitions and conventions.
- o Consistent user-interface conventions and procedures via a tool extension paradigm: a set of standards for keyboard semantics (control -C always does the same thing), error handling, help messages, menu management, forms management, etc.

4.4.4.1.3 Definition of PDSS Baseline Requirements Discussion

The Software Support Environment sub-panel reviewed several documents with the intent to extract useful portions to define an initial PDSS baseline requirement. The documents surveyed included:

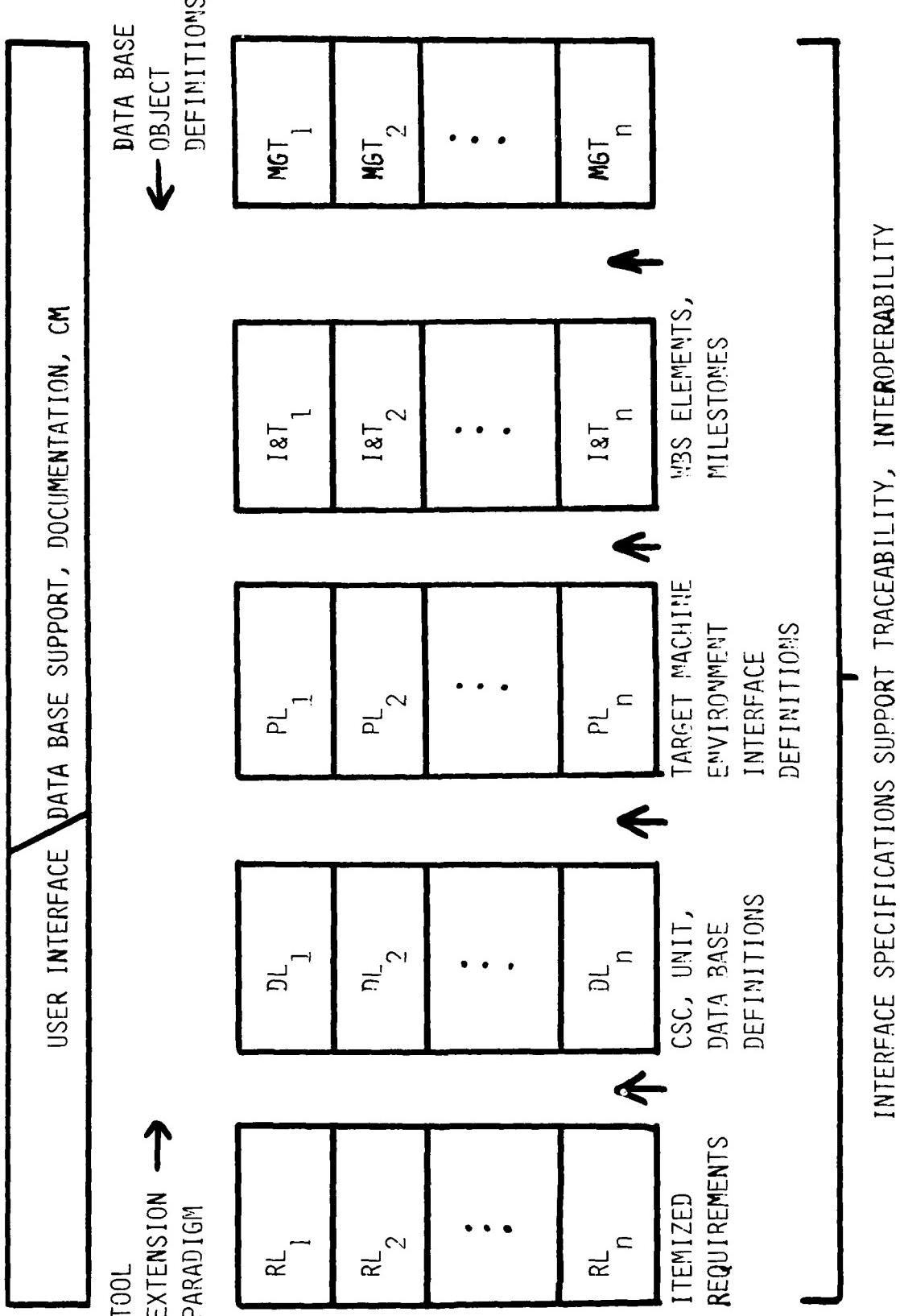


Figure 4.4-6 Accommodating Non-Generic Tools in a Generic Framework via MIL-STD-SDS

- o Software Engineering Automation for Tactical Embedded Computer Systems (SEATECS) Top Level Requirements, 31 August 1982, Naval Ocean Systems Center
- o Air Force Integration Support Facility, no date, Sacramento Air Logistics Center
- o Long Range Plan for Embedded Computer Systems Support, October 1981, TRW Corp.
- o A Software Engineering Environment for the Navy, 31 March 1982, Software Engineering Environment Working Group, Naval Material Command

Review of the documents began only after several days of work had been devoted to definition and agreement on:

- What is a PDSS?
- What might a PDSS physically look like?
- What are its functional characteristics?
- What portions could be made generic?

The evaluation of the documents was heavily weighted towards those that displayed a philosophy compatible with the SSE views which evolved from the discussions. See the following paragraph (4.4.4.1.4) for detail on this discussion.

The investigation found the SEATECS document mapped well to the SSE views and provided a subset of specific detailed requirements for the PDSS. (There were omissions in certain areas, e.g. system test, external communication, etc.) The format of the requirements as listed in the document also seemed appropriate for a top level requirements definition.

The panel decided to map the SEATECS requirements to the PDSS functional capabilities in order to determine what additional requirements must be added. This mapping plus the added requirements are defined in section 4.4.5.1.4.

4.4.4.1.4 PDSS DESCRIPTION

- o SCOPE. The Software Support environment must be designed to facilitate all functions of the PDSS as outlined below:

- a. GENERAL: A PDSS is an engineering facility established for the evolutionary support of mission critical software.

The PDSS facility is made up of people, generic equipment, physical environmental and communications facilities, data, documentation, and procedures. These facilities provide the capability to perform software development, simulate the operational environment, evaluate digital systems or subsystems, test software, integrate hardware and software, and address man/machine interfaces. PDSS facilities also provide the capability to maintain a configuration "baseline" and manage activities and configuration items developed in the facility. (see Figure 4.4-7).

- b. CAPABILITY AND FEATURES: The PDSS facility should have the following features and capabilities (see Figure 4.4-8):

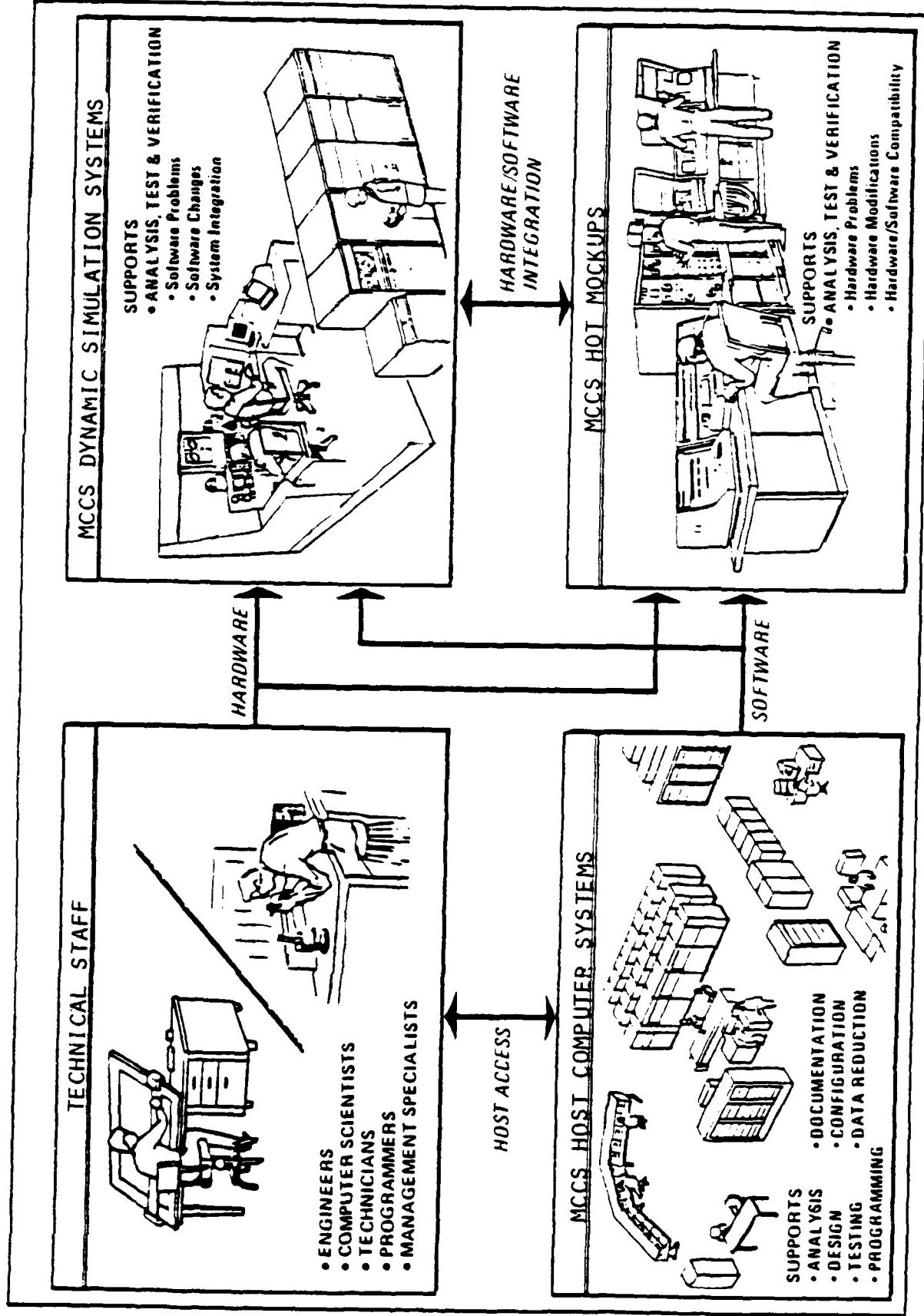


Figure 4.4-7 PDSS Facility

MCCS SOFTWARE SUPPORT

- Investigate & Resolve Problems
- Develop Changes
- Integrate & Test Baselines
- Evaluate Performance
- Independent Verification & Validation

HARDWARE SUPPORT

- Investigate Problems
- Evaluate Performance
- Develop Modifications

SYSTEM SUPPORT

- Integrate, Test, & Evaluate
 - Functional Performance
 - Hardware/Software Compatibility
 - Man/Computer Interfaces
 - Procedures
- Modes & Submodes of Operation
 - Mission Scenarios
 - Communication Interfaces
 - Tactics

MICROCOMPUTER SYSTEMS SUPPORT

- Software Development & Changes
- Hardware Development & Fabrication

Firmware Programming
System Integration, Test,
and Evaluation

- 1) Support mission critical systems within an integrated facility, including systems with multiple computers;
 - 2) Support multiple functions with common modules;
 - 3) Support harmonious interconnections of systems with dissimilar architectures, languages, program structures, and input/output requirements;
 - 4) Support extension and reconfiguration, as the number of systems within the PDSS are increased or decreased;
 - 5) Support mission critical software evolution through preplanned product improvements;
 - 6) Support life cycle management and systems engineering cost objectives;
 - 7) Incorporate existing support assets into the PDSS design and throughout its life cycle;
 - 8) Meet physical, technical, and administrative security requirements;
 - 9) Support rapid response mission critical reprogramming requirements.
- c. SUPPORT TOOLS: To enhance the productivity of PDSS personnel these facilities should have a set of standard tools with pre defined interfaces to allow inter-tool communication. These tools should be capable of supporting specific PDSS processes, as well as in concert with generic support processes. Automated PDSS tools should be included to support the following functions (see Figure 4.4-9):
- 1) Data Base Management.
 - 2) Configuration Management.
 - 3) Information Management.
 - 4) Requirements Analysis.
 - 5) Specifications Definition.
 - 6) Software Design.
 - 7) Documentation.
 - 8) Software Development.
 - 9) Software Test, Test Data Collection, Reduction and Analysis.
 - 10) System Integration.
 - 11) Simulation.
 - 12) Validation and Verification.
 - 13) Project and Cost Management.

These tools must be provided in concert with high order languages and their automated support environments for the management, development, re-engineering, test and integration of mission critical software.

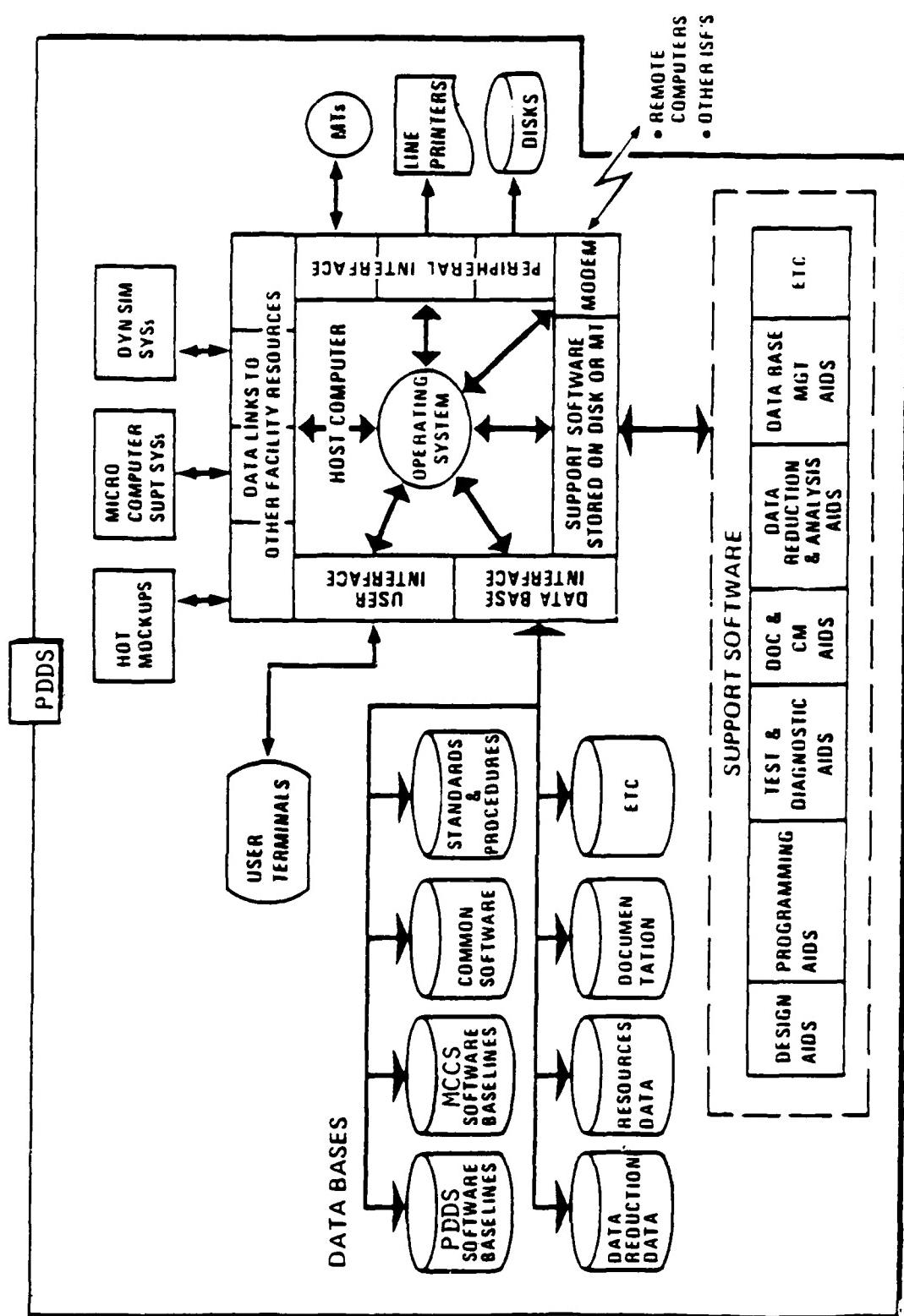


Figure 4.4-9 Ideal PDSS Host Computer System

d. PHYSICAL FACILITY: To prevent premature technical obsolescence, the PDSS facility should have an open core design, raised floors, relocatable walls, floor to floor chases, environmental and electrical distribution systems. Building flexibility into the initial design will allow the PDSS facility to be reconfigured to meet future tasks. The electrical system must meet the requirements of commercial computer systems, avionics and unique weapon systems. Where security is a requirement, all power must be filtered. Various special requirements must be met based on specific applications, such as requiring geographical benchmarks, special fire protection, special grounding or security needs. General PDSS facility requirements include:

- 1) Office space for engineers, computer scientists and support personnel,
- 2) Local area and long haul network nodes,
- 3) Power and cooling equipment for mission critical hardware systems and PDSS support systems,
- 4) Vaults for security and physical protection,
- 5) Libraries for magnetic and printed media,
- 6) Equipment maintenance areas,
- 7) Growth/storage space, and
- 8) Security planning.

e. SECURITY PLANNING REQUIREMENTS: Requirements for ADP system security mode and the security classification of the PDSS will be determined by the classification and use of the system being supported and the operating requirements. Planning must include sufficient lead time to comply with the approval requirements of cognizant agencies. (For additional data on PDSS security see paragraph 4.4.4.2.2.)

4.4.4.2 Subpanel 2 - Management Support Systems Considerations

Subpanel 2 discussed these basic areas:

- o Management Support Systems
- o Security Implications/Requirements for SSE
- o Advantages and Disadvantages of a CFE an/or CFE support environment

4.4.4.2.1 Management Support Systems

Management support systems for a generic post deployment support system environment have a set of unique requirements in addition to the requirements of a management support system for a generic development environment. The requirements unique to the PDSS stem from the requirement to manage the reproduction, distribution, implementation, and trouble reporting of multiple versions of the software product. In addition, for certain types of mission-critical systems the final portion of operational testing may only be accomplished when the entire system is operational; for example, a command and control system which supports the National Command Authorities and requires the transfer of information between service organizations and the unified and specified commands. Following are some areas of consideration for requirements for PDSS management support systems.

a. System Test

Specific tools may be required to perform system testing, capture system test results, and to perform system test result analysis. The availability of these tools becomes critical when the system supported interfaces with a system, (or systems) supported by a different PDSS activity.

b. Security

Because of the nature and use of the deployed software and system there may be implications on the security category of the PDSS and the trustworthiness of the software support environment software which is not inherent in the development process. An example might be where a non-classified program fails when utilizing classified data and the program can only be recreated with classified data. These factors impact personnel security, physical security, ADP system area controls, ADP system access controls, and transmission controls. Tools are required for monitoring and enforcing security especially in the area of ADP system access controls. These requirements may be more rigorous than those for the development ADP systems.

c. Deployment Monitoring

Information must be collected regarding the distribution of the product to where it is to be installed. This information must be readily accessible for ad hoc queries. For example, if the product is distributed via registered mail, information of the following type must be collected: date of shipment; contents of the shipment, registry number or numbers of the shipment, date of receipt, person signing for receipt, etc.

d. Operational Installation Monitoring

Information must also be collected regarding the installation of the product in the specific operational facility (e.g., aircraft by tail number). Again, the information must be readily accessible for ad hoc queries.

e. Trouble Reporting and Analysis Systems

Although the development process requires the collection of similar information, the amount of data processed for deployed systems is usually larger and the reporting procedure itself may be more complex due to the dispersement of the product. Considerations should be given to timely reporting, timely feedback to the reporting installation as to receipt of the trouble report, and analysis results. Consideration should be given to providing trouble reporting information to all installations and on-line trouble reporting.

4.4.4.2.2 Security Implications/Requirements

The ease of implementing security requirements in a software support environment is directly related to the environments rehosting philosophy. This is particularly true in the case of the current government efforts to develop Ada programming support environments (based on the use of a wide variety of commercial computers and associated (vendor-supplied) operating systems). The basic philosophy is to isolate machine/operating system dependencies to one easily managed and controlled area (KAPSE) with the major portion of the environment being machine/operating system independent (APSE). Rehosting of the APSE then only requires changes to the KAPSE to satisfy unique machine/OS dependencies of the new host.

The difficulty in designing the APSE to meet ADP security requirements is made more complex because of this rehosting philosophy. Because the security of the SSE involves the identification and control of both SSE users and SST information, it is clear that the sophistication of the system resources available to manipulate these data plays a large part in the ease or difficulty in implementing security measures.

Also, these resources are typically implemented in the machine itself, or the operating system, or both. It follows that, ideally, security measures should be implemented in the machine and/or operating system, as opposed to the APSE. However, the majority of existing machines and operating systems that will be used to host the APSEs have not been designed and developed to meet security requirements. The security measures required for a specific SSE must, therefore, be implemented within the APSE (assuming no changes will be made to vendor-supplied operating systems used to host the APSEs). The difficulty in developing "secure" APSEs, based on the present APSE rehosting philosophy, is readily apparent. One solution is to develop a security "interface" to isolate the APSE (secure) from the host operating system. Obviously, this can be accomplished by "encrypting" secure data that passes from the APSE to the host operating system, and de-crypting the data as it passes from the host operating system to the APSE. Equally obvious is the (potentially) extensive overhead (performance, space, etc.) necessary to implement this security interface.

4.4.4.2.3 Provision of the Software Support Environment

a. GFE

A uniform GFE Software Support Environment has the potential of significant total life cycle cost savings.

- o Common hardware (CPU's, peripheral equipment and special equipment) can be purchased at less cost due to economies of scale.
- o A common SSE will reduce the numbers of SSE's in the DoD inventory, reduce SSE ownership costs and result in more effective use of scarce personnel resources.
- o A common GFE environment implies that similar operational procedures could be used at the PDSS's. This means that continuity of procedures and operations could be easily maintained across different MCCS's within a PDSS and from PDSS to PDSS. Also, workload leveling can be accomplished between separate PDSS's during peak or critical times at each center.
- o A common environment will tend to reduce both hardware and software maintenance costs for that environment (for example, an operating system change could be installed and verified on a central beta test PDSS and then deployed to all PDSS's). Training both government and contractor personnel in the use of the PDSS hardware and software (task) is simplified. Unified training programs could be developed and given at each PDSS.
- o The generic SSE lends itself to open bidding. There would be no proprietary data (contractor) or hidden technical expertise requirements.
- o The government would buy all source and data rights.

b. Non-GFE

Factors related to a non-GFE environment are:

- o The "standard" SSE implies some obsolescence because of its necessarily longer life.
 - Contractors would have to merely accept (and charge for) inefficiencies inherent in a generic SSE.
 - If a contractors newly developed tools were distributed to all future developers there would be no incentive for industry to develop new tools.
 - There are not many examples of non-GFE PDSS SSE's, except perhaps for the defacto standard VAX computer with VMS or UNIX.

- It is difficult to define an environment that would satisfy the needs of the developer. The government has limited experience in this particular area.
- o Higher project costs could result from the potential inefficiencies of a GFE SSE. (The reverse could also, of course, be true, e.g., lower project could result.)

c. GFE/Non-GFE Mix

Is it feasible/realistic to define a generic SSE tool set? What is Generic? A generic environment is defined as being:

- o Hardware independent.
- o Language independent.
- o Specific tool independent - This is a major issue related to generic environments. The services desire that to the fullest extent feasible the exact set of tools used in the original development of the software for the MCCS be used in the PDSS of that system. Industry however desires that the PDSS environment be defined (i.e., data base format, types of information required, interfaces to tools, etc.), and that the contractor be required to only ensure that developed code be supportable on the specified PDSS environment.

d. General Discussion

Sub-issues in this area which require further consideration are:

- o The complexity and multiplicity of MCCS's make it very difficult to envision one PDSS environment that is capable of supporting multiple systems.
- o The burden is on the initial RFP and proposal writers/approvers to require support details, assurances, and risk probability on a defined "specific" PDSS environment. Can and will this be accomplished?
- o Technically, can contractors/vendors assure supportability on a PDSS environment that is similar in functionality but which differs in exact tools?
- o Application independent criteria which must be considered when defining the PDSS requirements.

4.4.4.3 Subpanel 3 - Contractual Provisions

The Contractual Provisions subpanel of the Support Software Environment Panel focused on two major areas. The first area was the nature of the environment

(both development and the PDSS) in which major new focuses are seen as "project data bases" and "testing process" development. The second area dealt with the "environment, acquisition and use" in which discussion and guidance was postulated in "acquisition of the environment," "user aids for the environment" and specific unique "PDSS tool needs." A discussion of each of these topic areas follows.

4.4.4.3.1 Project Data Base

The project data base is the collection of information generated as the software evolves from requirements to termination of support. It is not to be confused with the system software data base which may be included in the project data base. The purpose of the project data base is to provide machine manipulation of the tremendous volume of information necessary to economically support the evolution of the system software throughout the life cycle.

The support activities, including the developer, must have common accessibility to the data base, either by use of common host computer and tools, or by the specification of the data base and the procurement of the necessary manipulative tools. It is the opinion of this subpanel that this data base possesses the same attributes throughout the life cycle differing only in the emphasis and focus. It is recognized that the data base evolves during the initial development effort.

The data base should be consistent with the specification (e.g., MIL-STD-SDS) used for the initial procurement and with the PDSS facility to be assigned final responsibility (if known). A DID needs to be developed to provide guidance in the definition of the contents of the data base. Sources for the initial set of elements for use in the DID are referenced in Appendix B - Miscellaneous. In addition, the DID should consider the requirement for the data base to include module definition, module interfaces and support reference tables. The data base shall support the automation of traceability from requirements through design to implementation.

The tools for the manipulation of the data base shall include the ability to access by a selected parameter(s) in a timely manner. The traceability features shall be bidirectional and shall facilitate the evaluation of the impact of proposed changes and facilitate the design and development of new capabilities in the MCCS. The data base shall be such that it can be economically validated against the implemented MCCS. The data base must include the intra-/inter-system interfaces providing the description of modules, subroutines, tests, data sets, etc. These elements are critical to the effective management of the software structures.

The specification of the project data base will shift the key support elements from the traditional delivered documents. The traditional documents are development tools and those portions of these documents which are provided in the project data base are the elements of the life cycle support. The traditional documents are still necessary to insure that the understanding of the software is correct and current. Configuration control of this data base is required.

A properly designed project data base and related support tools will provide improved configuration management capabilities, simplify the quality assurance process and provide significant management insight into the status and effectiveness of the MCCS throughout the life cycle.

It is emphasized that the data base must be automated. That is, the tools necessary to access and manipulate the data base to provide the requisite engineering and management information in a timely manner are critical to the efficient support of MCCS, both during development and at PDSS.

The documentation for supporting the tools is important, for the documentation should include that information which is necessary to provide adequate 'maintainability' and also user information sufficient for PDSS personnel to effectively use the tools.

The procuring activity shall ensure that the project data base is specified for delivery to the PDSS. If the data base environment and/or tools are not GFE or GFI (I = Information), then the data base must be completely defined and accurately specified. The tools required for the manipulation of the data base must be specified and the PDSS environment clearly defined such that the data base and tools will be utilized to the PDSS personnel as delivered.

4.4.4.3.2 Testing Processes Development and History

Testing is the primary method for insuring high quality software. Different types of testing are performed during the life cycle. Additionally, the thoroughness of the testing is a variable since it is impractical to perform 100% testing; i.e., exercising all combinations of data states and program paths. Therefore, it is recommended that:

- o During acquisition the type and thoroughness of the testing must be explicitly stated in the contract and all documentation and test data, specifically test inputs, test directives, and test results, shall be delivered in the project data base. The PDSS facility shall have all tools, or their equivalent, necessary to duplicate and repeat the acquisition testing.
- o The PDSS facility shall have the additional tools and capabilities necessary to perform the remaining types of testing to an arbitrary degree of thoroughness.

4.4.4.3.3 PDSS Environment Acquisition

It has been asserted that the environment required for the PDSS is largely identical and in fact a superset of the project development environment. This assertion is true in either an Ada environment or a non-Ada environment (which the PDSS world must continue to support for multiple decades). Appendix D attempts to portray an assessment of one major PDSS facility with some gross estimates of focus shift between development engineering focus and the PDSS

focus. Note that in gross summary the functions are similar, but in detail functions the PDSS is presented with added management and performance optimization requirements which in fact do add new support requirements and associated tools and procedures (hence the superset).

Given that the PDSS environment is at least equal to (possibly a superset of) the development engineering environment, the following question must be posed: How can the acquisition process insure that the development environment (or its functional equivalent as a minimum) is available for the PDSS facility and can be effectively integrated into the operational doctrine of the intended PDSS?

The following paragraphs address the issue: The first and the easiest solution is to direct the developer to use GFE/GFI facilities and environment either located at the PDSS or a copy of that environment which is or will be added to the PDSS.

- The Naval Air Development Center's Facility for Automated Software Production (FASP) is one example.
- PDSS capability comes early.
- Ownership of the environment and its process and procedures reside with the PDSS or its parent agency.
- New human interfaces need not be developed (familiarity).
- Fewer environments are needed (divergent proliferation for convergent purposes).

The second approach is to provide GFI the environment and supporting system to the developing agency mandating their use and an acquisition agency interface to the process for monitoring and development sequence control. In this case, the environment is essentially specified by the GFI package - ownership of which resides with the Government. PDSS facility preparation can be carried on in parallel and probable PDSS readiness for turnover is enhanced.

- The Navy's Machine Transferable Support Software (MTASS) is an example of a limited environment.
- Current GFI packages are limited in capability.
- Environment development and maintenance is already in long term planning vice new dollar requirements.
- Control of procedure is nearly equal to GFE approach.
- Cost is reduced.

The third approach which supports the PDSS objective is to allow the developer to use an environment that is neither GFE nor GFI (or possibly an extension of a GFI), but rather some commercially available or licenseable environment set which is compatible with a proposed PDSS facility. In this event the acquisition agent must acquire the identification, full specification and system dependencies of the environment and insure that sufficient resources are in-place at the proposed PDSS to support the environment in use. Basically, this approach will install a new environment at the PDSS which is a copy of the development environment. This installation accompanied by necessary training and augmentation of PDSS resources as appropriate must be done early (12-16 months before turn-over) to insure effective PDSS operations.

- RSL Revs is an example.
- IBM PDS II at Trident is an example.
- Does not interfere with developer's desires.
- Requires careful monitoring to control environment stability.

The fourth option is to allow a developer to use a proprietary internalized environment that is optimized for the developers institutional processes. In this event, the acquisition agent must insure the establishment of formal turn-over points for the project data base, the data base format, content and the support tool functions which are critical to the development sequence of the project (i.e., Macro P Procs, table drum dB generators, etc.).

In all of the above approaches there are three principles that must be followed by the acquisition agent.

- o Require control points consistent with the environment in use through the development of the project. Through the data base, "automatically" validate the data base against the appropriate level of systems implementation and plan the traceability of the system requirements through the current level of system implementation.
- o Require the cost of PDSS preparation, whatever option is selected, to be an evaluation factor in the total acquisition of the system. In this way, an efficient new development environment, which is not part of the PDSS current capability, can be fairly offset against the cost of upgrading the PDSS facility and vice versa.
- o Mandate ownership, transportability or rights in use, or licensing and life cycle support of proprietary software used in the development environment and which will support PDSS operations. Ignore all other proprietary software.

4.4.4.3.4 PDSS Reference Manual

The acquisition agent should acquire during the development phase a documented design approach to the PDSS operation called a reference manual. The purpose of the PDSS reference manual, and of possible supplemental training courses, is to convey to the PDSS staff specific knowledge that the developers have that will make it easier to maintain and enhance the software system. It should be required as a contract deliverable in addition to the standard required specifications, etc. It is not intended to further specify the delivered software; it is an operationally-oriented guide specifically addressing the activities to be performed by PDSS personnel in maintaining the software.

The outline found in Table 4.4-1 describes the types of material the PDSS manual should include. Other organizations may be equally appropriate, and additional contents may be desirable, depending on the particular situation; the outline is intended to convey a general view of the manual's function.

It may also be desirable to procure training courses covering the same kinds of subject matter. This decision depends on several factors, including complexity of the system, quality of the documentation, and initial availability of enough of the PDSS staff to make training worthwhile. The PDSS manual is considered essential; the need for additional training is discretionary.

4.4.4.3.5 Tools Which are Unique To The PDSS

The PDSS activity requires the basic set of software development tools used by the development activity - they are not enumerated here. In addition, PDSS activities require additional tools that may not be provided in the development environment. The procurement activity should ensure that these tools are acquired.

The need for additional tools in the PDSS stems from the different orientation of the PDSS; its primary orientation is to respond to change. Hence additional tool requirements derive from the need to manage, control, analyze the impact of, and adapt to changing requirements. Specific tools that these functions require are:

1. Configuration Control Tools
 - a. Tools to manage multiple versions and releases.
 - b. Tools to distribute (e.g., perhaps by electronic means) multiple versions and releases.
 - c. User information release and management tools.
2. Impact Analysis Tools
 - a. Tools to query project data base(s) to assess impact/magnitude.
 - b. Tools to assist in cost assessment.

Table 4.4-1

PDSS MANUAL

OUTLINE

1. INTRODUCTION

1.1 Purpose and Scope

1.2 Assumptions

- What level of knowledge and training is assumed for users of the manual?
- What tools and equipment are assumed to be available at the PDSS in order to carry out the procedures described.

1.3 Reference Documents

- Should include relevant software design specs. tool and methodology manuals, etc.

2. Environment Summary

This section gives an overview of the environment to be used in maintaining the software and lists and briefly summarizes applicable tools, data base, applicable methodologies, etc. It does not replace the documentation for these items, which should exist independently. It should reference this documentation.

3. Software System Summary

This section gives a brief summary of the software system to be maintained. It is not intended to replace any of the software specs, and should reference them for detail. It should include:

- Brief discussion of what the software does (functional summary).
- Top-level structure of the software (what components; how they fit together).
- Description of the hardware configuration.

Ideally it should provide a roadmap for using the system and software specifications.

Table 4.4-1 (cont)

4. Project Data Base

Assuming that an online, automated project data base is used, this section should describe it. If the data base is separately documented, a summary and references to the document are sufficient. Otherwise, this section must document the specific contents and use of the data base.

If the project data base is not automated, this section of the manual should contain (or reference) the on-paper information that takes its place. This includes tabular information such as traceability matrices, interface lists, etc.

5. Dependencies on Other Software

This section documents any known dependencies that the software system has on other software. In particular, this includes:

- Operating system (OS) dependencies, if the software makes use of a vendor-supplied OS.
- Compiler dependencies. (Everything is dependent on the specific code generated by the compiler; this section should highlight aspects expected to be particularly vulnerable to new compiler releases.)
- Interoperating systems.

The best presentation here would present a "cookbook" approach to what to do when a new release of the OS, compiler, or whatever is received; i.e., what to look for, what is likely to change, etc.

6. Dependencies on Hardware

This section contains practical advice for adapting to hardware changes. The hardware in question would include the processor(s) on which the software executes as well as any peripheral devices or equipment with which it interfaces.

6.1 Processor Dependencies

If the processor is upgraded or modified, what are the likely impacts?

If the processor is replaced by a different kind of processor, what are the likely impacts? (This assumes the use of an HOL such as Ada. While programs may be generally portable, they will contain processor dependencies that must be documented here.)

Table 4.4-1 (cont)

6.2 Device/Equipment Dependencies

This subsection should enumerate each device that the software interfaces with, identify the software component(s) that contain the dependency, and discuss what action should be taken if the device is modified, eliminated, augmented, or whatever is likely.

7. Performance Maintenance

A continual function of the PDSS is to maintain software performance as the system evolves (if not to improve it initially). This section should present:

- processor utilization data
- known critical areas (i.e., "any change here should be done very carefully -- it will have major performance implications")
- description of any available measurement/tuning tools and techniques
- hints for further performance tuning (perhaps things the developer knows about but didn't get to)

8. Memory Utilization

Another PDSS problem is keeping the software in memory as it grows and evolves. This section should assist with that problem by providing:

- memory maps
- description of any overlay or memory management strategies in use
- known "break points" that would permit additional overlaying, off-loading to another processor, or whatever
- any other hints as to how space could be saved (e.g., tables that could be compressed, in-memory data that could be purged, etc.)

9. Envisioned Changes/Enhancements and How to Make Them

Many of the changes and enhancements that might be required for a given software system can be predicted. This section should identify these and give "cookbook" procedures (insofar as possible) for responding to them. This would include such things as:

Table 4.4-1 (cont)

- how to add an "x", where "x" is a new, e.g., communication protocol, user terminal, radar type, whatever
- how to expand capacities
- how to alter limits, protections, etc.
- how to change anything that's been parameterized
- how to adapt to an enhanced user interface (e.g., graphics terminal replaces teletype)
- how to extend security (probably impossible unless it's been planned)

10. Functional Scenarios

The environment/tool manuals document the use of the tools. This section documents sequences of actions the PDSS personnel would perform to accomplish particular functions. For example, such scenarios might include:

- how to process a user trouble report
- how to conduct a system regression test
- how to evaluate a change proposal

These scenarios would explicitly step through the tools and procedures required.

11. Additional Maintenance Hints

This is the place for the developer to provide any other advice he may consider useful.

3. Management Control Tools

- a. Multiple subsystem resource planning and management tools.
- b. Additional management and utilization reporting tools (because there are additional reporting requirements in the PDSS environment).

4. Performance Maintenance Tools

- a. Performance measurement/monitoring tools.
- b. Optimization tools.

5. Testing Tools

- a. Tools to determine which tests test which system functions (and hence which must be redeveloped or re-executed if the function changes).
- b. Tools to determine which tests test which modules (and hence which must be re-executed if the module changes).
- c. Path coverage analysis tools.
- d. Automated regression test tools.
- e. Test results comparison (i.e., to expected results or to previous results).
- f. Data collection, reduction and analysis tools.

6. Simulation Tools

- a. Interoperability/external system interface simulators/stimulators.

4.4.4.4 Subpanel 4 - Application Area Unique Criteria

4.4.4.4.1 SSE Requirements.

A generic software support environment (SSE) or a post-development software support (PDSS) facility could be established to support all types of DOD software including operational software, ATE software and training software. The establishment of a generic SSE is possible because the basic components that comprise an SSE are the same for any SSE. Differences between the environments come from differences in emphasis of the environment towards ATE, training or operational software. This difference in emphasis results in an environment that differs in specific hardware make-up, personnel make-up, size and complexity. For example, an SSE to support ATE software will have the same types of components as one which supports operational software but it may have different compilers, computers, support software, etc.

However, it is generally impractical to develop a fully generic SSE with the capability to support all types of software. The impracticality could be caused by facility size restrictions which would cause the ineffective use of resources. In these cases grouping the software support into PDSS facilities could be accomplished by function such as Electronic Warfare (EW), avionics, training, communications, etc., by weapon system/platform such as aircraft, ship, etc., or by specific hardware such as radars, etc. The selection should be accomplished based upon economies of scale.

4.4.4.4.2 Operational Environment Considerations

The intended system operations environment (air, land, sea, or space) of the system being supported does not by itself place special or unique requirements on an SSE (or PDSS facility). However, the target computer and associated interfacing hardware must be available in (or available to) the SSE for integration and test regardless of the operational environment.

4.4.4.4.3 Impact of Generic SSE on Resources

A generic SSE serves to share resources. The generic environment reduces the amount of support software to be maintained, can improve the sharing of personnel resources, and enhances transportability and interoperability. The generic SSE concept can be carried too far. If the generic concept grows too large, management of resources will become too complex and conflicts of priority will arise.

A generic SSE will serve to consolidate support software. Many support software tools can be effectively utilized across many programs within the SSE. These tools might include such items as automated configuration management documentation, and program management/status software packages. These items would be more fully utilized if spread over many programs. In addition, use of these tools would tend to form a "standard" which would increase efficiency and productivity.

A generic SSE can serve to improve the use of personnel resources. Personnel expertise in specific areas can be used across many programs. In addition, sub-specialties can be developed to allow more effective utilization of employees when workloads shift. There are two potential drawbacks to the use of these personnel. First, is the danger of expertise being spread too thin which would cause a lack of responsiveness to requirements. In addition, management of large numbers of personnel could cause an unwieldy bureaucracy to grow which would not be responsive to PDSS needs in a timely fashion.

A generic SSE could enhance transportability and interoperability of software. Depending upon the systems being supported it may be possible to develop software packages that are used in multiple systems. Packages such as digital communications, weapon control and tactics, and tracking algorithms may be examples of such multiple use. In addition, if the systems being supported must be interoperable on a digital communications net, the sharing of communications software packages may enhance the interoperability of such systems.

4.4.5 CONCLUSIONS AND RECOMMENDATIONS

4.4.5.1 Subpanel 1 - Definition of the Support Software Environment

4.4.5.1.1 Environmental Architecture View

Reference: Paragraph 4.4.4.1.2.2

This SSE architecture and concept of operation implies the following:

- Conclusion: The commonality factors and economics of scale of a single unified SSE for a referenced system outweigh the advantages of having several SSEs supporting different phases of the software life cycle.
- Recommendation: PDSS environments for defense systems should utilize a single unified architecture such as the one indicated in Fig 4.4-3.

4.4.5.1.2 Life Cycle View

Reference: Paragraph 4.4.4.1.2.3

- Conclusion: Other than the distribution management functions required for PDSS, there are no significant differences between the generic functions required for a development mode SSE and a post development mode SSE. Any SSE function which provides effective support in one mode also provides effective support in the other mode.
- Conclusion: The development SSE and the post development SSE are almost identical. Primarily, the post development SSE must grow to support distribution management and many defense system functions and capabilities not identified during development.
- Recommendation: The CRLCMP¹ should identify the directions of evolution which the SSE will need to support, identify the organization responsible for supporting the post development evolution of the SSE, and provide a clear transition plan between the development SSE and the post-development SSE.

4.4.5.1.3 Other Perspectives

Reference: Paragraph 4.4.4.1.2.5

The current state of the art in understanding the nature of interface standards and conventions is incomplete, leading to the following:

¹ Computer Resources Life Cycle Management Plan

- Conclusion: From a DOD ownership standpoint, it is extremely important to establish interface definitions between components of the Software Engineering Environment (SEE) to promote commonality, interoperability, and evolution among SSE's. However, the current state of the art does not support a complete definitive specification of these interfaces.
- Recommendation: The current STARS SEE effort should develop an initial definition of these interfaces, conventions, and paradigms, and support further R&D toward a more complete definition.

4.4.5.1.4 Definition of PDSS Baseline Requirements Discussion

Reference: Paragraph 4.4.4.1.3

Recommendation: We recommend that the additional requirements items be added to the basic SEATECS document, and that document be adopted as the starting point for the Top Level Statement of Requirements for the PDSS.

Products: The system architects view of the PDSS consists of a number of subsystems upon which functional requirements can be mapped. Table 4.4-2 illustrates the panel's suggested mapping between the SEATECS requirements and the PDSS subsystems. Requirements not included in the SEATECS document are indicated by an asterisk and are defined after the table. A key to the tables entries is found at the end of the table.

NOTE: It is recommended that the Navy's SEEWG and STARS SEE documents be examined to determine if they rather than the SEATECS document should be revised.

4.4.5.2 Subpanel 2 - Management Support Systems

4.4.5.2.1 Security Requirements

Reference: Paragraph 4.4.4.2.2

Recommendation: PDSS centers should analyze their security requirements and obtain a host computer with the security features that support those requirements. As a minimum the level of certification of the host computer should be known. This means that proposed host systems should have been evaluated by the Computer Security Evaluation Center.

Further, it is recommended that the JLC sponsor a study of long term PDSS security requirements with emphasis on Ada run-time environments, and other PDSS unique requirements.

4.4.5.3 Subpanel 3 - Contractual Provisions

4.4.5.3.1 Project Data Base

Reference: Paragraph 4.4.4.3.1

Table 4.4-2

PDSS/SEATECS MAPPING

<u>PDSS Subsystem</u>	<u>SEATECS Functional Requirement</u>
Kernel OS Service	3-1.9; 3-4; 3-5; *P-1
Command Interpreter	3-1.1 thru 3-1.8
User Interface	3-2.1; 3-2.2
DBMS	2-1 thru 2-6; 3-3
Utilities	1-10.1; 1-10.8
Tools/Programs/Documentation/Mgmt	1-1; 1-2; 1-3; 1-4; 1-5; 1-6; 1-7; 1-8; 1-9; *P-2
Communications I/F	1-10.9; 1-10.10; *P-3
H/W Test Drivers/Target Systems/ Instrumentation	1-7; *P-4
<u>P-1</u>	
Allow multilevel security access to the PDSS data base.	
a. Discussion/Example: The PDSS will have multiple types of users. Some users require extensive access to the PDSS data base while others require minimal access. The data base also has different security classifications which must be protected.	
b. Criticality in PDSS Environment: Productive	
c. Technological Feasibility: Experimental	
<u>P-2</u>	
Support Distribution of software releases to operational sites/systems/ agencies.	
a. Discussion/Example: The system will interface with or be a part of the overall CM system so that site addresses, configurations, etc., are joined to aid in or mostly automate the software distribution process.	
b. Criticality: Immediate	
c. Technological Feasibility: Available	
d. Implementation Priority: I	

Table 4.4-2 (cont)

P-3

Support and interface to all commonly used communications standards and protocols.

- a. Discussion/Example: Download developed software to a target system via MIL-STD-188C (or other) interface.
- b. Criticality: Immediate
- c. Technological Feasibility: Available

P-4

System Test Capability

- a. Discussion Example: A system mock up must be provided that will allow easy final system test. The mock up should include support devices that permit software download, test monitoring (break and trace), data collection, results comparison, and environmental simulation.
- b. Criticality: Immediate
- c. Technological Feasibility: Available

KEY

- o Criticality is defined as:

Immediate. Capabilities which current support systems provide; capabilities without which the most basic mission cannot be performed effectively.

Productive. Capabilities which would have notably positive effects on productivity; capabilities without which the PDSS can function initially but which will become necessary as the workload increases.

Desirable. Capabilities which will enhance the PDSS's ability to perform their mission but whose absence will not be detrimental to mission performance.

- o Technical Feasibility is defined as:

Available. Capabilities which can be purchased off-the-shelf today and with little or no adjustment be usefully employed in an SSE for the PDSS's includes capabilities requiring rehosting.

Table 4.4-2 (cont)

Convertible. Capabilities which are technologically available today but which do not precisely fulfill PDSS needs, thus requiring some addition, modification, rewriting, retargeting; includes more adequate engineering, control, and/or support for operational use.

Experimental. Capabilities which exist today in a limited and/or experimental automated form whose use is still considered to entail risk; includes incompleteness of theory and incompleteness of implementation; could be introduced into an SSE for use by those projects which understand and could accept the potential risk.

RECOMMENDATIONS

All MCCS acquisitions be required to include the project data base and associated tools in the software development process.

Develop a Data Item Description to define the content of the project data base and the minimum set of manipulative capabilities required.

Require the project data base to be automated. The magnitude of the information to be accessed in the project data base is such that manual techniques would not be effective.

Develop management guidelines and procedures for the effective use of the project data base. Include Configuration Management, Quality Assurance, and Verification and Validation use of the project data base.

4.4.5.3.2 PDSS Environment Acquisition

Reference: Paragraph 4.4.4.3.3

RECOMMENDATIONS

Modify current directives to formally recognize the types of environment development as described in paragraph 4.4.4.3.3.

Change acquisition policy guidance to mandate the three principles developed as described in paragraph 4.4.4.3.3.

Incorporate in MIL-STD-SDS, or other acquisition guidance the concepts of project data base delivery/program control and the test process and history and the PDSS reference manual approach.

Maintain a clear focus of the unique performance shift between the development facility and the PDSS task even though strong similarities now require requisite tools and environment exist. Tools that exist in a PDSS that exceed the nominal functional requirements of a development facility are:

- Multiple version release/distribution management.
- Performance monitoring/analysis/optimization tools.
- Extended test tools (i.e., regression).
- Interoperability/external system interface stimulators.
- Impact analysis/change assessment.
- User information release and management.
- Multiple subsystem resource planning and management tools.
- Management and utilization report variances (controlling to different principles).

4.4.5.3.3 PDSS Reference Manual

Reference: Paragraph 4.4.4.3.4

Recommendation: JLC consider adding the PDSS Reference Manual DID to the post-development support upgrade of MIL-STD-SDS. See Table 4.4-1 for the recommended manual format.

4.4.5.3.4 Tools Which Are Unique to the PDSS

Reference: Paragraph 4.4.4.3.5

RECOMMENDATION

Insure that all of the necessary unique PDSS tools are acquired during the development phase as well as the development environment data bases needed.

4.4.5.4 Subpanel 4 - Application Area Unique Criteria Recommendations

Reference: Paragraph 4.4.4.4

RECOMMENDATION

Generic PDSS environments should be established to take advantage of common resources necessary to perform PDSS. The generic PDSS concept should be established based upon logical groupings of systems requiring support. This logical grouping should be based upon the principle of economies of scale. PDSS should be large and broad enough in scope to allow effective utilization of existing resources and be responsive to support needs. Yet if the PDS environment becomes too large it may not be able to effectively respond to the support needs of the supported systems.

The JLC (or other appropriate agencies; e.g., DCA, USCG, etc.) should periodically review PDSS facilities and the systems supported by these facilities to assure that systems are supported in the most responsive and economical manner. Of particular interest in this review would be the systems supported: Do they all belong here? Is there a system (or systems) supported here that could be better supported at some other facility? Such a review would provide recommendations for consideration by the individual services, commands, or agencies.

PDSS facilities should be encouraged to communicate with each other in order to share success stories and lessons learned. This communication could also lead to the sharing of tools, techniques, and practices.

APPENDIX A
PANEL D
PARTICIPANTS

Co-Chairs

Jim Hess - U.S. Army
Jerry Raveling - Sperry

Subgroup 1

Howard Klien - TRW
Barry Boehn (Co-Chair) - TRW
Jim Winchester - Hughes
Oscar Staudt - USAF

Dave Boslaugh (days 1-2) - USN
Dan Burton - USAF
Ajmel Dulai - USAF

Jim Williams - ROLM* *To subpanel
Bob Sacks - GRUMMAN* 2 on Wed PM
Joe Batz - DOD* and Thurs

Subgroup 3

Chris Braun - SOFTECH
Hank Stuebing - USN
Skip Meiers - USCG (Chair/Recorder)
Russ Eyers - USN

Subgroup 4

Brian Goldiez - U.S. Army
Ed Tognola - U.S. Army
John Martinsen (Chair/Recorder) - Boeing
Frank Campbell - IDA
Ed Williams - USN

Al Patterson - USAF
Pat Haverty - Burroughs (SDC)
George Sumrall (Co-Chair) - U.S. Army

Subgroup 2

Caral Gianno - DCA
John Cole (Chair) - U.S. Army
Owen McOmber - USN
Bob Sauer (Recorder) - USMC

PANEL D - S/W SUPPORT ENVIRONMENT

CO-CHAIRMAN:

Hess, Mr. J. (Jim)
HQ DARCOM/DRCDE-SB
5001 Eisenhower Avenue
Alexandria, VA 22333
(202) 274-9318
A/V 284-9318

Replacement For:

Lieblein, Dr. E. (Ed)
Director, CCS, DUSD (R&AT)
Room 3D139 (400 AM) Pentagon
Washington, DC 20301
(202) 694-0208
A/V 224-0208

CC-CHAIRMAN:

Raveling, Mr. J. (Jerry)
Sperry Corporation
Computer Systems, M.S. U1E13
P.O. Box 43525
St. Paul, MN 55164
(612) 456-3545

MEMBERS:

A	Sumrall, Mr. G. (George) CENTACS/DRSEL-TCS Ft. Monmouth, NJ 07703	(201) 532-1004 A/V 992-1004
A	Tognola, Mr. E. (Ed) AVRADA/DAVAA-SS Ft. Monmouth, NJ 07703	(201) 544-4201 A/V 995-4201
A	Cole, Mr. J. (John) CECOM/DRSEL-TCS-SIO Ft. Monmouth, NJ 07703	(201) 544-2759 A/V 995-2759
A	Goldiez, Mr. B. (Brian) PM. Trade, DRCPM-TND-EM Naval Training Equipment Center (50C) Orlando, FL 32813	(305) 646-5761 A/V 791-5761
N	Stuebing, Mr. H.G. (Hank) Naval Air Development Center Warminster, PA 18974	(215) 441-2314 A/V 441-2314
N	McOmber, Mr. O.L. (Owen) HQ, Naval Material Command (MAT 08Y) Washington, DC 20360	(202) 692-3966 A/V 222-3966
N	Eyres, Mr. R. (Russ) Naval Ocean Systems Center San Diego, CA 92152	(619) 225-7069 A/V 933-7069
MC	Sauer, Mr. R.E. (Bob) Marine Corps Tactical Systems Support Activity Camp Pendleton, CA 92055	(619) 725-2617/2618 A/V 993-2617/2618
AFLC	Patterson, Mr. A. (Al) SMALC/MME McClellan AFB, CA	(916) 643-6316 A/V 633-6316

PANEL D (CONTINUED)

AFSC	Burton, MAJ. C.D. (Dan) USAF ESD/ALL Hanscom AFB, MA 01731	(617) 861-2002 A/V 478-2002
AFSC	Dulai, Mr. A. (Ajmel) ASD/AXT Wright Patterson AFB, OH 45433	(513) 255-5941 A/V 785-5941
AFSC	Staudt, Mr. O. (Oscar) AFCCP/C/SKXD Tinker AFB, OK 73145	(405) 734-7145 A/V 735-7145
CG	Meiers, CDR E.J. (Skip) USCG U.S. Coast Guard 5307 Foxborough Ct. Alexandria, VA 22310	(301) 871-3615
D	Giammo, Ms. C. (Caral) Defense Communications Agency/JDSSC 11440 Issac Newton Sq. N. Reston, VA 22090	(703) 437-2338 A/V 364-2338
D	Batz, Mr. J. (Joe) OUSD (R&E), Pentagon, Rm. 3D1079 Washington, DC 20301	(202) 695-7756 A/V 225-7756
A	Winchester, Dr. J. (Jim) Hughes Ground Systems Group P.O. Box 3310, Bldg. 618 Fullerton, CA 92634	(714) 732-5576
A	Braun, Ms. C. (Chris) Softech 460 Totten Pond Road Waltham, MA 02154	(617) 890-6900
A	Boehm, Dr. B. (Barry) TRW Systems Group 1 Space Park Redondo Beach, CA 90278	(213) 393-4206
A	Campbell, Mr. F. (Frank) (IDA) Institute for Defense Analyses 1801 N. Beauregard St. Alexandria, VA 22311	(703) 845-2284 A/V 289-1948
N	Sacks, Mr. R.M. (Bob) Grumman Aerospace Corp. Mail Stop B38-035 Bethpage, NY 11714	(516) 575-7255
N	Martinsen, Dr. J.L. (John) Space & Mil. Appl. Div. Federal Systems Group, Boeing Computer Services P.O. Box 24346 (M/S 9C-24) Seattle, WA 98124	(206) 575-5149

PANEL D (CONTINUED)

AFLC	Klein, Mr. H. (Howard) TRW P.O. Box 1058 North Highlands, CA 95660	(916) 920-2613
AFSC	Williams, Mr. J. (Jim) Rolm Corporation MSC Division (M/S 110) 1 River Oak Pl. San Jose, CA 95134	(408) 942-7657
O	Haverty, Mr. J.P. (Pat) Burroughs Corporation 2500 Colorado, Blvd. Santa Monica, CA 90406	(213) 820-4197
NTEC	Williams, Mr. E. (Eddie) Naval Training Equipment Ctr. (N-401) Orlando, FL 32813	(305) 273-6184 A/V 791-4111

APPENDIX B
BIBLIOGRAPHY

DoD Directives

DoDD 5000.29, "Management of Computer Resources in Major Defense Systems"
26 April 1976*

DoDD 5000.31 (Draft), "Computer Programming Language Policy"
June 1983*

Regulations and Instructions

AFR 800-14, Volume I, "Management of Computer Resources in Systems"*

AFR 800-14, Volume II, "Acquisition and Support Procedures for Computer Resources in Systems"*

AFLCR 800-21, "Management and Support Procedures for Computer Resources Used in Defense Systems"*

DARCOM-R-70-16, "Management of Computer Resources in Battlefield Automated Systems" 25 May 1982

SECNAVINST 5200.32, "Management of Computer Resources in the Department Navy Systems," 1 March 1979

NAVAIRINST 5230.9, "Policy and Procedures for the Establishment and Operation of Naval Air Systems Command Software Support Activities," 14 June 1983

Manuals

DoD5220.22M, "Industrial Security Manual"

Studies/Reports/Papers

Hdqtrs. U.S. Army, "Post-Deployment Software Support (PDSS) Concept Plan for Battlefield Automated Systems," May 1980

TRW, "Air Force Embedded Computer System Support Improvement Program," October 1981

Naval Material Command, "A Software Engineering Environment for the Navy," 31 March 1983

Naval Ocean Systems Center, "SEATECS - Software Engineering Automation for Tactical Embedded Computer Systems - Top Level Requirements," 31 August 1982

* Revision in progress

APPENDIX B
PAGE 2

DoD, "Software Technology for Adaptable, Reliable Systems (STARS) - Support Systems" 30 March 1983

AFLC, "Air Force Integration Support Facilities; Their Total Utility," Paper by Alton E. Patterson, Sacramento Air Logistics Center (NAECON 80)

Military Standards

MIL-STD-SDS (Draft), "Defense Systems Software Development Standard"
August 1983

DoD-STD-1679A (Navy), "Software Development"
22 October 1983

ANSI/MIL-STD-1815, "Ada Programming Language"
February 1983

Miscellaneous

NOSC Technical Note 932, The Project Development Data Base: The Core of an Automated Software Engineering Environment, Patricia A. Santoni, Code 8324, dated October 1980

NADC, Facility for Automated Software Production Methodology Handbook

APPENDIX C PANEL PRESENTATION SUMMARIES

This appendix provides a brief summary of the briefings which were made to the Panel on Tuesday, November 1, 1983.

- o "A Builder's Guide to Software Engineering Environments"
Mr. William E. Riddle, Software Design and Analysis, Inc.

Mr Riddle discussed:

- o What is a software engineering environment?
- o What tools could be provided by a software engineering environment?
- o Why bother getting and using a software engineering environment?
- o How can a software engineering environment be obtained cost effectively?

The briefing served to provide:

- o An overview of environments and appropriate environment terminology.
 - o An understanding of the various environments that currently exist.
 - o A more detailed description of several existing environments.
 - o Arguments on the value of environments.
 - o A discussion of how to plan for constructing an environment from purchased and self-developed parts.
- o "A Modern Facility for Software Production and Maintenance"
Mr. H.G. "Hank" Stuebing, Naval Air Development Center, Warminster, PA

Mr. Stuebing described the Facility for Automated Software Production (FASP) which has been designed, developed, and is in use for life-cycle support of weapon system software. This facility consists of a software system which runs on a commercial multicomputer configuration at NADC. For additional detail on FASP, see Appendix D of this report.

- o "Electronic Warfare Avionics Integration Support Facility (EWAISF)"
J. J. (John) La Vecchia, AFLC EW Management Division, Robins AFB, GA

Mr. La Vecchia provided an overview of the EWAISF in terms of its:

- o Purpose
- o Scope of Responsibility
- o Physical Facilities
- o Hardware Structure/Processor Network
- o Languages and Computer Architectures Supported
- o Basic Change Process.

The EWAISF supports all USAF airborne electronic warfare equipment including radar receivers, jammers, and tactical reconnaissance systems.

APPENDIX C
PAGE 2

- o "United States Army Post Deployment Software Support (PDSS) Study"
J. "Jim" Hess, DARCOM, Alexandria, VA

Mr. Hess discussed the results of a recent U.S. Army study on PDSS. The report developed a plan for Army PDSS for battlefield systems. The plan identified eleven (11) PDSS centers and recommended policy to improve the PDSS environment both before and after deployment.

The briefing discussed the purpose of the PDSS centers, background of Army battlefield system support, alternative approaches investigated by the Army, and the status of the current Army PDSS approach.

- o "A Software Engineering Environment (SEE) for Weapons System Support"
H.G. "Hank" Stuebing, Naval Air Development Center, Warminster, PA

Mr. Stuebing discussed the current Software Technology for Reliable Adaptable Systems (STARS) task entitled, "Definition/Preliminary Design of Software Engineering Environment." This task, begun in June of 1983, has the objective of developing a Joint Service definition of a SEE. A SEE is an integrated system that supports embedded computer software over the entire life cycle. The common definition will assure the interoperability of tools while offering the advantages of an integrated environment.

APPENDIX D
A SOFTWARE ENGINEERING ENVIRONMENT (SEE)
FOR WEAPON SYSTEM SOFTWARE

H.G. Stuebing

Naval Air Development Center

Warminster, PA

A SOFTWARE ENGINEERING ENVIRONMENT (SEE) FOR WEAPON SYSTEM SOFTWARE

H.G. Stuebing
Staff Consultant
Software and Computer Directorate
U.S. Naval Air Development Center
Warminster, Pennsylvania 18974
United States of America

SUMMARY

A Software Engineering Environment (SEE) has been designed, developed, and used for the life-cycle support of weapon system software. This SEE consists of two types of facilities; software production and integration. The software production facility consists of a software system that runs on a commercial multicomputer configuration. The approach features increased management visibility of the software development process, increased programmer productivity through automation, reducing the cost-of-change during maintenance, and the use of automated regression testing to improve software quality.

These facilities have been used for seven years to develop and maintain weapon system software for several projects. This paper describes accomplishments, refinements to the code and test functions, and a general approach to extend the capabilities into the requirements and design phases. Techniques are described that simultaneously allow different methodologies, programming languages, and target computers to be implemented on the same host computer. Also discussed is the implementation of a SEE in a distributed computer network.

1. INTRODUCTION

"Software engineering" is concerned with developing software systems that satisfy the requirements of the user over the life of the system; a SEE assists the accomplishment of software engineering through sets of computer facilities, integrated software tools, and uniform engineering procedures. The term "weapon system software" inherently implies a concern with software for embedded computer systems and support over the entire life-cycle.

A generic view of the weapon system software life-cycle phases is shown in figure 1. This figure emphasizes the view that weapon system software is redeveloped several times during maintenance, the time after the initial version is delivered. The development process has overlapping phases, each with a measurable input and output. The phases overlap to show that there is an interaction between them. Within each phase, a set of activities is defined to systematically achieve the goals; the functions of management, quality assurance, and configuration management are included as activities in each phase. There are iterations horizontally, between activities of a particular phase, and vertically, between phases. During initial system development the work progresses through all phases. During maintenance the point of re-entry is determined by the scope of the intended change.

At the U.S. Naval Air Development Center (NADC), Warminster, Pennsylvania, facilities have been constructed to assist software engineering for weapon system software. Two types of facilities were built: software production and integration. The Integration Facilities were built for each project and consist of laboratory hot-mockups of the embedded computers with realistic simulation of external inputs. The software production facility is an integrated software environment hosted on a large-scale commercial multicomputer configuration. The host configuration consists of five Control Data Corporation (CDC) mainframes, a CYBER 175, CYBER 720, CYBER 780, and two CDC 6600's. This software production facility is called the Facility for Automated Software Production (FASP) and it is described in (STUEBING, H.G., 1980) and (FASP, 1979). The conceptual and architectural ideas of the FASP were strongly influenced by (BAUER, F.L., 1971) and (SOFTECH, INC., 1974). The FASP became operational in July 1975 and was the first integrated environment to be used for weapon system software and among the first integrated environments. The FASP supports the activities shown in figure 1 from Mission Requirements to Code and Test; however, only the Code and Test phase is supported by an integrated environment. In the earlier life-cycle activities the support is provided by loosely-coupled sets of tools, an important distinction discussed later.

2. ACCOMPLISHMENTS

In this section the accomplishments of the FASP are discussed; the reported period of operation is July 1975 through March 1982, nearly seven years. These accomplishments are given to set the context for the discussion that follows and to give encouragement to those who are contemplating establishing such facilities regardless of scale. In judging these accomplishments one must consider the methods that were used before the introduction of the FASP. Generally, before July 1975 the software development was done on the target computer itself. Sometimes the same target computer was used for development and integration. The state of support software and peripheral devices for these

target computers was primitive compared to the state of commercial computers. Nevertheless, a large industrial-based work force had established a way of "doing business" with these facilities and produced large amounts of weapon system software. The software problems of those days are well documented.

2.1. Integrated Environment Hosted On Commercial Computers

The FASP experience has shown that the concept of an integrated environment hosted on large-scale commercial computers can be used as a true production facility. True production means that the availability and performance are adequate to produce weapon system software. The FASP was contractually specified as government-furnished equipment with guaranteed performance; the usage was almost entirely contractor personnel located at remote sites. A table of data is shown in figure 2. The data shows in summary form some of the key parameters measured with the FASP. The amounts of software shown are larger than the software delivered to the fleet since some projects keep several versions active at any given time. Also, the data refers to on-line software and does not include the amount of archived software.

This approach used a one-time development of support software that not only eliminated such tasks from the contract but also used less in-house personnel than supporting separate facilities for each project.

2.2. Management Visibility And Control

The FASP provided the dual functions of an advanced programming system and a management information system; this allowed management visibility into the software development process at a detailed level. Data base controls allowed configuration management to be enforced from the beginning of projects, a welcome benefit. Also, the FASP was a natural way to have work standards uniformly enforced over a group of projects. The NADC was able to serve in various roles with regard to weapon system projects; these roles included System Prime, Validation and Verification, and Life-Cycle Support Activity. The FASP, with its remote terminals, could be used by contractors or other government laboratories regardless of geographical location.

An advantage of having software developed in the FASP is that there was no transition required when the project was transferred to the maintenance phase. Further, since the software was in a government operated facility with all management information, test data, and documentation in hand, the maintenance of the software could be competitively procured in a realistic way, a major change from the times of being captured to one vendor for the life-cycle.

When software was not developed in an integrated environment it was found to be poorly organized and impossible to recreate without the original developers. The development of interface control documents proved invaluable when such software had to be transferred to the FASP from another development facility. When the developed software did not follow the interface control documents, the effort to transfer the software was sometimes large. Transition efforts took a few days when the interface standards were followed and varied from a one-half of a man-year to seven man-years when they were not.

2.3. Productivity

In (STUEBING, H.G., 1980) the productivity, measured in delivered source lines per man-month, was well over 400. This data was measured before significant interactive features were added to the FASP. The data was a two-fold increase over published industry data for real-time embedded computer software; the productivity data is believed to be greater with interactive features. The data before the FASP was sparse and was not consistently measured. There are local examples of turnaround time varying from one to several days with target computers being used for the development facilities. These times also do not include the courier travel time to and from the facility. With the FASP the turnaround time is measured as viewed from the remote terminal. The FASP speed improvement and the complete elimination of courier travel time reduced turnaround time by a factor of twenty.

With the FASP we have had examples of sharing large amounts of software between projects. The sharing is made easier because both projects are in the same facility and it is simply a matter of copying a data base. Further, the use of a common facility by a large group of people tends to result in better communication among the group as a natural byproduct.

2.4 Quality

The quality of the software produced by the FASP is significantly better than previously. Some of the reasons are: comprehensive unit testing with software emulators; enforcement of standards; better tools; and improved management visibility into the software development process.

Consider the following example. The FASP was being used to support a Verification and Validation effort on weapon system software that was developed by a contractor in a separate facility. An interface control document was in effect and the software was scheduled to be delivered as functional increments, each delivery adding to the previously delivered software and giving additional functional features. Figure 3 shows a plot of the delivered source lines (not counting

comments). This software was quickly installed in the FASP and subjected to many unit tests, using the FASP in the regression testing mode with path coverage analysis. As new deliveries were received, additional tests were added and run with all the previous tests. Figure 4 shows the number of software errors that were recorded after each delivery; at the end of the effort 89% of all paths were tested. Figure 4 is significant for several reasons.

First, before the delivery to the FASP the contractor believed the software to be suitable for fleet use; however, the contractor had not used an integrated facility with regression testing or path coverage analysis to test the software.

Second, the errors were reported to the contractor while the development team was still in place and the resultant error data was a factor in the computation of the contract award fee. This resulted in the correction of many of the errors.

Third, all the error corrections were done before fleet delivery. Clearly, this was more cost effective than waiting until the errors were reported from the fleet.

2.5. Technology Transfer

The FASP has been used to support projects in several of the Naval System Commands for airborne, surface, and sub-surface applications, a much broader scope of use than originally expected. This usage has all been on the central computer facilities at the NADC.

The entire FASP software system was successfully transferred to a major aerospace corporation. They plan to use it for all Navy software that they develop. Today, the full FASP system is only portable to other CDC CYBER computers.

A version of the FASP was rewritten using the UNIX operating system on the Digital Equipment Corporation VAX 11/780 computer system (UNIX is a trademark of Bell Laboratories). This version of the FASP supports several popular microprocessors; components that are appearing rapidly in weapon systems. It is planned to bring the UNIX version to the same level as the CYBER version, forming a product that will be easily portable to many other users.

3. "TO THE SEE"

3.1. The Software Problem

The hardware (the physical embedded computer resources of the weapon system) is generally considered less of a problem than weapon system software. The "software problem" has been covered extensively in the literature; however, different aspects of the problem have been emphasized over time.

In the early 1970's there was great concern over the quality of weapon system software. The performance, reliability, and user-friendliness were poor; most errors occurred during coding and remained undetected after testing and integration. This condition has significantly improved through better design methods and comprehensive testing; most errors are now traceable to erroneous requirements, not coding.

Today's paramount issue about software is productivity; that is, the achievement of a true economic increase in productivity over the life-cycle, (MORRISSEY, J.H., 1980) and (MUNSON, J.B., 1981). It is well known that software is a labor-intensive field and that the life-cycle costs are both high and rapidly increasing. For a given weapon system about 25% of the software life-cycle costs are for development and 75% for maintenance. Software productivity, in an economic sense, has only increased modestly when measured over the life-cycle. Most of the available labor is devoted to maintenance and the amount is rapidly rising because more and more systems are being deployed. The demand for labor with software skills has exceeded the supply, a trend expected to continue through the 1980's. As the balance of labor continues to shift to maintenance, less and less labor is available for development. Therefore, to reverse this trend in the future, it must be cheaper and faster not only to develop software but also to change it during maintenance.

A SEE consists of sets of computer facilities, integrated software tools, and procedures that support a weapon system over the life-cycle. A SEE serves as a unifying element to assist software engineering and forms a basis for attacking the software problems of quality and productivity. Testing remains the primary method for assuring software quality; a SEE can provide many automated aids to minimize the labor required for testing. Productivity is improved with a SEE not only by automating the testing of software but also by aiding all steps in the development process, including making it easier to reuse large amounts of software.

Each phase of the life-cycle employs different engineering methods. Within each phase there is usually a choice of several methods for each activity. The term "methodology" refers collectively to a selected set of engineering methods. Ideally, one methodology consisting of uniform methods would exist to cover the life-cycle. The transitions between phases would be smooth as well as the transitions between activities of a particular phase. An ideal SEE would be highly integrated, both horizontally and vertically. However, such an ideal state is some time in the future. Therefore, the issue of today is choosing a way to evolve toward the ideal SEE; it is a matter of implementing what is practical while continuing research into improved methods.

A key to success is to create a framework where new tools and techniques can be continually superimposed on existing work activities in a nondisruptive manner. Further, it is better to allow multimethodologies, to the extent possible, than to attempt to select the one "true" methodology. For example, in the Code and Test phase instead of building facilities that implemented only the Chief Programmer Team approach, it would be better to chose a way that allowed several methods, one being the Chief Programmer Team. The activities concerned with Code, Test, and Integration are better understood than the remaining phases; therefore, they offer a natural starting point. There is considerably more variability to the methods and techniques currently available in the requirements and design areas; thus, a loosely coupled collection of tools is more appropriate for those phases.

3.2. SEE Versus Programming Support Environment (PSE)

Several terms have appeared in the literature that are similar to "SEE." They are: programming environment, programming support environment, and software environment. These terms have been generally used to describe the Code and Test activities, although frequently mention is made to the requirements and design phases. Further, these terms have usually been restricted to the software concerns of a system and not the system as a whole. Therefore, the distinction is that the term "SEE" is more general and includes the above terms. A SEE refers to the support over the life-cycle including aspects other than purely software.

Of course, the term "environment" itself can be somewhat confusing in this context. This term is so general it is difficult to determine its limits in some texts. In this paper the term refers to the "work environment" for the phases of the weapon system life-cycle. The emphasis is on the facilities that support the work of each phase and the interface between the engineer and the computer. It is recognized that organizational and social factors are an important part of the work environment. These factors must be considered in the design of any computer-based support system but they are not the main points discussed in this paper. An important point is to recognize that the engineer's detailed view and use of the support facilities is different depending on the phase of the life-cycle. The needs are different between coding, unit testing, and integrating the weapon system software with the embedded computer. Likewise, the detailed view of the software is different between "development" and maintenance. The term "meta-environment" has been used to describe the aspects of environments that depend on the user's view of the system and the organizational and social setting (ELZER, P.F., 1979).

In industrial engineering there has been considerable work on facilities, both in concept and implementation. There are many valuable observations in this field that can be applied in part to a SEE. However, there are also some important limitations. For example, in industrial engineering the production facilities are oriented to rapid and automated replication of physical devices, devices that have been previously designed. Thus, there is an area called Computer-Aided Manufacturing (CAM) where the computer has been applied to the task of automating the production of physical devices. Separately, there is an area called Computer-Aided Design (CAD) where the computer has been applied to assisting the designer, making, for example, integrated circuit layouts. In the software field the work of coding and testing bears a similarity to production in the industrial engineering sense. This was the main theme in the development of the FASP. Now as we attempt to extend these facilities into the requirements and design phases it is important to note that the analogy must be to "CAD" and not to "CAM." In these phases the facilities must support both the cognitive processes of the designer and the more clerical aspects of recording the results of the processes. Therefore, the theme of the SEE in the early phases is to assist the engineer during the cognitive processes and to automate the clerical aspects of recording information and generating documentation.

3.3. An Integrated System

The term "integrated system" is also frequently used in the literature. Now, the dictionary definition of "integrate" is clear: "to make whole or complete by adding or bringing together parts, to put or bring parts together into a whole; unify." Thus in creating an integrated system, the designer would do tradeoffs between the parts to achieve a unified whole. A SEE is referred to as an integrated system; it appears to the user as a unified whole that assists the accomplishment of software engineering. The users include both engineers and managers; the work activities vary over the life-cycle and the user interface and capabilities vary accordingly. Conceptually, the SEE may be considered as a single entity that presents to the user different views and capabilities according to the phase of the life-cycle. The implementation is most likely to be several computer-based facilities that have similar user interfaces but different specific capabilities depending on the phase of the life-cycle.

The two types of users, engineers and managers, means that their individual needs must be tradedoff such that the final system represents a unified whole. The software engineers need advanced programming capabilities for the Code and Test phase and the managers need relevant, consistent information, and a means to control the cost and schedule of the effort. The software engineers need compilers, linkers, system generators, and simulators to accomplish their work; these components are called "tools." The managers need a means of identifying the end-items that are to be built, a way of monitoring progress, and a way of insuring that the agreed-on procedures are being followed. The data base concept is a natural way of collecting such management information; it also provides a way of meeting many functions of the engineer.

Clearly, if an engineer were given just a tool set, the work could be completed. If the tools were compatible with one another, forming an integrated set, the work would be easier to accomplish. However, such a tool set usually relies on the host operating system and the engineer is thus free to create any set of files that is felt to be appropriate. If

many engineers are working on the same project, the chances of them all retaining the same type of information in their files is small. In such a case the manager has no easy way to determine what end-items are being produced or what progress has been made. Of course, this hypothetical team could agree to follow a set of standards; but changes in personnel, deadlines, design changes, etc., would quickly destroy the good intentions. Also, the end-item productivity of this team is lower because experience has shown that a large part of their energy is diverted to handling the file system and writing support programs.

The computer is the natural and convenient place to integrate the needs of the engineer and manager and provide an integrated system to accomplish the work. A SEE is an integrated system in the above sense; it is much more than a tool set, it is a unified system that meets the needs of both engineers and managers.

4. CODE AND TEST

In this section the characteristics of an Integrated system to support the Code and Test phase are briefly described (STEUBING, H.G., 1982) contains a more detailed description). The system that is described is not specifically the FASP, but one that has been generalized and refined based on seven years of operational experience with the FASP. During the seven-year period, three major evolutions took place along with extensive feedback from the users.

4.1. A Dual System

The system should provide the dual functions of an advanced programming system and a management information system. The needs of the manager must set the top-level framework of the system. This requires a selection of the engineering methods and procedures and deciding what information should be saved; it implies choosing a method or allowing only certain methods to be supported by the system. The methods must have a sound engineering basis and fit the organization's business methods. With the FASP it was a conscious decision to support several methods with the same system. The FASP facility is owned and operated as a government facility and used by weapon system contractors to develop and maintain software. Each contractor had different methods and procedures for doing business, yet each was able to effectively use the FASP. Some contractors have used the Chief Programmer Team approach, others a different team approach. All use some form of structured programming, although the details are different. Thus as a government facility, it was important to impose only reasonable constraints on the contractor and allow for the different ways of doing business.

An important concept in software engineering is incremental development. The idea is to first complete the software design and then to build the software in stages, or increments, such that each successive increment adds a new functional feature. This approach breaks the work into smaller pieces that are easier to manage. Thus, it is easier to judge progress on the total project and it has the additional benefit of allowing users to gain some early experience with the software system. Incremental development has proven to be valuable on large-scale software projects and should be supported by the SEE.

An important management need is the enforcement of configuration management principles. Configuration management principles consist of identification, control, status accounting, and the establishment of baselines. Decisions must be made regarding what software elements will be subject to configuration management. This involves deciding what is the smallest unit of software that will be configuration managed. Is it a "line" of code? Is it a "module"? Is it the basic compilation unit of the compiler? Is there more than one language to be supported and is the definition of a compilation unit the same for both? These decisions have a significant impact on the final system and must be made at the outset.

Along with structured programming came the idea of include segments. Include segments are fragments of code that are used in many modules without change. The programmer identifies these segments by name and places them in the data base; in the source code of a module the segment is referenced by name. The system automatically locates the segment and "includes" it into the source stream before compilation. Since these segments are fragments of code, they cannot be separately compiled without errors; however, they are useful to the programmers. In the FASP during a typical month the data bases contained 48,000 modules and 28,000 include segments, showing the wide acceptance and use of include segments. Therefore, although include segments add to the configuration management burden, they are recommended for a SEE. Another related area is access control. Management level decisions are needed regarding what aspects of the system are subject to access control. In the FASP there are three dimensions of control; control over access to the software end products, control over software tools, and control over access to the computer system itself. In the latter case this implies cost control over the use of the computer system.

4.2 The Data Base

The data base is the most critical component of the SEE since it serves as the unifying element for all other components. The data base contains not only the weapon system software but also related technical and management information that contains the genesis and status of the total effort. Furthermore, the data base has a significant influence on the performance of the SEE, an important consideration in obtaining a true production environment.

Here the term data base refers to a fixed number of libraries that are encapsulated and managed as a whole rather than distinct parts. The software for a particular weapon system is contained in several "data bases." Each data base contains the following libraries:

- The source library, containing either the source code for modules or the source code for include segments, or both;
- The object library, containing the object code corresponding to the source library;
- The test library, containing test input data, previous test results, test directives, and system generation directives;
- The interface data library, containing information such as linkages to external object programs or to shared source code;
- The production data library, containing modification histories, and a variety of management information; and
- The documentation library, containing all documentation about the weapon system software.

The encapsulated data base is the basic unit that the SEE deals with. Several libraries are included because the relationship between those libraries must be strictly enforced. Thus, source and object code must have a one-to-one correspondence with no exceptions. A consequence of this relationship is that if compilation errors occur the data base (source and object libraries) will not be updated! Likewise, test data and test results are synchronized. Most important, at any time in the development schedule the management data is consistent with the rest of the data base; thus, managers always have access to accurate information.

An important feature of the SEE for large-scale projects is the automatic recompilation of dependent modules when certain software is modified; for example, if an include segment is modified then all modules that use that segment will be automatically recompiled.

Commands are available that allow software to be shared between the data bases, allowing the total effort to be divided among several data bases and teams. Similarly, commands allow data bases to be divided into smaller ones or combined into larger ones. Other commands allow the data bases to be copied.

The integrity of the data base must be assured; therefore, during interaction with the SEE the system automatically creates a backup copy of the data base permitting instantaneous fall-back to the previous version. Additionally, commands allow archive copies to be made on magnetic tape for off-line storage. This level of protection is over and above that offered by the host operating system.

4.3 Procedures, Tools, Commands, And Processing

The use of the SEE involves sequences of tool and data base interactions. To simplify the use of the system a set of procedures is defined that are invoked by user commands. A procedure is a set of computer directives that automates a particular work task, invokes the proper tools in the proper sequence, provides all data base manipulations and correspondences, and automatically records statistics of all activities.

The software tools are programs that do certain functions for the software engineer. Examples are: editors, translators (compilers and assemblers), system generators, test analyzers, software emulators (target computer instruction level simulators), data extractors, report generators, and documentation aids. A tool or set of tools are automatically invoked as part of the execution of a procedure. Some tools are visible to the user, such as the editor, and require communication in a language unique to the tool. Other tools are invisible to the user, such as the librarian, and are automatically invoked when certain actions take place with the data base. In the latter case input and output data may be processed by other programs but all such actions are hidden from the user.

User commands cause procedures to be invoked. A command is a procedure name followed by parameter values. These values give the user flexibility in directing the procedure to accomplish the specific desired function. All commands are validated before being executed. The commands can be grouped into two categories, Immediate and Queued, depending on whether the data base is modified or not. In batch mode, there is no distinction and all validated commands are executed in the order received. In interactive mode an "Immediate" command is executed at once; all others are placed on a command queue. Once activated, the system executes the queued commands.

As an example of the power of commands, consider the FASP command Modify Software (MODSW). This command is used to create or modify software in data base. In the FASP on the CDC CYBER computers, MODSW causes 315 job-control-language commands to be executed; in the FASP on the VAX computer it causes 262 UNIX shell-script commands to be executed. These operating system level commands are all hidden for the user.

A general set of procedures has been developed and is described in (STEUBING, H.G., 1982). Figure 5 shows a list

of these general procedures. The procedures are divided into functional groups and are described by process flow diagrams. These diagrams use a structured English description of the control flow for a procedure and a data flow diagram showing the process performed, the tools used, the data base contents used and produced, and other information required by the procedure. When a procedure is performed a certain amount of standard processing is done before and after the main processing for that procedure. The standard processing for each procedure is shown in figure 6 using structured English. Two process flow diagrams are shown in figures 7 and 8.

4.4 Testing

Testing remains the primary method for determining the quality of software. The SEE should support four distinct types of testing during the code and test phase. They are:

- Progression testing that evaluates new or modified software operation;
- Regression testing that identifies changes to previously attained software operation;
- Automated test analysis that measures the effectiveness of a test by identifying the software source code paths exercised; and
- Trial testing that provides for testing proposed software changes without modifying the data base.

Progression testing is used during the development of new software or modifications to existing software. This form of debugging is frequently an intense creative process best performed interactively. It usually will involve interactive use of the software emulator to make experimental changes to initial conditions, data or instructions, and immediate rerunning of the test.

Regression testing is used once proper operation is achieved. It insures that the software does not deteriorate (regress) due to subsequent progression changes. Two forms of regression testing are used, explicit and automatic. In the explicit form the user specifies the tests that are to be performed. In the automatic form tests are automatically run whenever certain modules are modified. Test data, test results, and test directives are accumulated in the data base during the life of a module; also, an index is kept that relates tests to modules. A change to the module triggers the automatic running of all associated tests and a comparison of all results. The user is able to identify those portions of the test results that are important.

Automated test analysis is provided to check the quality of the tests themselves. In this form of testing a tool called Automated Test Analysis (ATA) scans the source code and inserts software probes at program decision points. This allows the decision-to-decision paths to be identified. When the instrumented code is run on the software emulator with the test input data, the system reports how many times each path was executed, flagging those not executed. Thus, the percentage of total paths tested is available along with indications of "dead code" and code paths most frequently executed. This data allows the user to devise changes to existing tests or to develop more effective tests. The data on the most frequently executed paths is valuable when optimizing the speed of the program.

Trial testing consists of syntactic and semantic checks before the software is entered into the data base. This type of testing is used when changes are made to large existing bodies of software. In such cases there may arise uncertainty about the interactions between changes to the software and to tests. Also, uncertainties about the optimum changes that could be made may require that several different changes be tried before deciding on the best.

4.5. Interactive/Batch

In many ways the differences between interactive computer jobs and batch have disappeared; however, there are some fundamental differences that are important to the operation of a SEE. A batch job consists of a stream of user commands with all parameters and input data previously determined; an interactive job must have these items supplied on-line. Since the system is to be user-friendly, the interactive job must prompt the user for such items and provide some helpful information when incorrect data has been input. Thus, the SEE must distinguish between the two types of operation and provide some extra software for interactive usage.

In a SEE for weapon system software it is unlikely that all the tools will be interactive, especially the compilers. Therefore, it is a matter of judgement to determine what functions are best performed interactively; all functions should operate in the batch mode. There are three areas that should allow both interactive and batch operation: they are editing, debugging with the software emulator, and generating management reports.

With the editor there is a clear benefit to the user to be able to quickly inspect and change the software. Full screen editors appear to offer the best advantages. When debugging, particularly during progression testing, there is also a benefit to the user. Here errors tend to be discovered more frequently and once observed the remaining parts of the tests can be terminated, saving computer time. Management reports, especially the smaller ones, tend to be the most useful when they can be quickly and easily obtained by the manager whenever desired.

Just as interactive mode is best for progressive testing, batch mode is best for regression testing. Here the total running time increases as the project software grows, a case best left for overnight turnaround when computers are lightly loaded and costs are frequently reduced. For example, when the FASP has been used for maintenance of large bodies of software the ratio of interactive-to-batch commands is about 3 to 1 on the average; however, in times of intense regression testing the ratio becomes 1 to 2. If one considers tool invocation during the same period then the ratio of interactive-to-batch is about 1 to 3 on the average; during intense regression testing the ratio becomes 1 to 12.

4.6 Multilanguages and Multitarget Computers

The design of a SEE is greatly simplified if there is only one programming language to be supported for a single target computer. For weapon systems this is rarely the case. Figure 9 shows the matrix of languages and target computers in the FASP. The difficulties begin with the languages themselves. A SEE is dependent on the programming language, a point not generally understood. One problem is the definition of a module is different in all the languages. Also, there are different dependencies on the data base between the languages. For example, if one language has structured programming constructs with an include segment feature built into the compiler, and another language does not, then clearly there is a significant difference in the way that a SEE would support each language. In the latter case it may be desirable to provide the capabilities by way of preprocessors; however, the way the SEE supported each language would still be different.

Other problems arise owing to structural differences in the languages. For example, the versions of the CMS-2 language have an order dependency on the appearance of declarative statements and executable statements. The declarative statements are dispersed throughout the program in blocks followed by blocks of executable code with an order dependency on the referencing of data. In this nightmarish state any change normally would mean that the entire program would have to be recompiled. However, in the FASP a special modular compilation feature has been added such that the system keeps track of the dependencies and only the appropriate blocks are recompiled. This example might appear extreme but it is characteristic of the difficulties that can arise.

With weapon system software there is likely to be several high-order languages and several assembly languages that must be supported. It is, perhaps, best to present the SEE to the user as an integrated string of tools that apply to a language and target computer. This would correspond to the vertical columns of figure 9. During the log-on process the user identifies the desired string, actions remain with that string because only those tools have meaning with one another. Internally, the system may use many common tools such as an editor or librarian; however, to the user the system appears as a unified set of tools.

The consequence of having strings of tools is that there are in effect several different SEE's. In the FASP there are four such systems concurrently operating in the host computers at any given time. From a maintenance standpoint, the FASP is maintained in a FORTRAN FASP. About 75% of the code is common across the four, the remaining unique to each language dependent environment. The language unique portions are maintained separately from the common portion and combined when a new version is desired.

4.7. Management

The success of a SEE depends on the degree that management is satisfied. Although a SEE brings many advanced tools to the user and makes the job of producing or maintaining software easier, it also constrains the user to work in a somewhat rigid framework, a point the user is sometimes quick to make. However, the benefits in productivity, improved quality, and stability over the life-cycle are great compared to any perceived loss of freedom by the user.

The degree to which management is satisfied depends to a great extent on the amount of involvement by management, the degree to which the system is understood by management, and how smoothly the SEE fits into the current methods of doing business. As with any management information system, the SEE requires that management become more involved with the operation at a deeper level than previously. However, once this commitment is made the gains are great.

The management view of the SEE is through the reports; therefore, it is desirable to generate clear, concise reports in terms that managers can understand. Reports that measure work progress and expenditures against planned profiles are interesting to management. For example, to report that the effort is on schedule and within funding regarding the number of modules, lines of code, storage size, and target computer execution time is obviously valuable to managers.

There is no general agreement across the industry on what precise software measures should be made; therefore, each organization must establish such measures and slowly refine them based on experience. It is important to allow a high degree of flexibility for SEE management reports.

Software complexity measures have been somewhat disappointing as absolute measures of software quality (PARISEAU, R.J., 1979). However, some are useful as relative measures and can be used for management control purposes. Generally, care must be taken in the selection of such complexity measures.

A promising area appears to be "earned values" reports and other related measures. These reports can be easily

established in a SEE and have the benefit of being based on impersonal data directly from the software development or maintenance environment. Of course, considerable experience is needed to select the particular "value" that is earned; however, there is a reasonable expectation that this can be accomplished.

5. THE EXTENSION TO REQUIREMENTS AND DESIGN PHASES

It has been stated that the SEE should support a weapon system over the entire life-cycle as shown in figure 1. It is intentional that the term "system life-cycle" has been used rather than "software life-cycle." Today, software is so important that it must be taken into consideration at the system level. Here the term "requirements" is used somewhat loosely to cover the phases in figure 1 from Mission Requirements to Software Requirements; perhaps, the terms system requirements and system design are more accurate.

The requirements phase begins with high-level statements about the mission of the weapon system. During a sub-phase called concept formulation a set of requirements is evolved that begins to express the requirements in technical terms. The activities at this point are not highly structured. The system designers used high-level tools such as analytic simulations and the methods of Operations Research to do tradeoff studies and to verify the conceptual design.

Once the system requirements are expressed in technical terms the system architecture must be determined in detail. The critical issue is the allocation of the system functions to hardware or software implementation. The system designers need tools to assist the tradeoff analysis. This activity is probably the most difficult of the entire process since the final system's cost and performance are largely determined by these allocations. Once the allocations are made, any changes become not only increasingly difficult but also increasingly expensive as the system moves toward operational deployment.

When the hardware and software allocations are completed the system development splits into two paths; the resultant hardware and software efforts come together at system integration time. The software requirements should be expressed in a formal requirements language so automated tools can analyze them for completeness and consistency, the two major sources of errors. At the end of the phase the requirements should exist in a computer data base so all formal documentation can be automatically generated.

The software design process begins with the formal software requirements and results in a specific software design. This design is a specification for the code. During this process the designers synthesize a software system that satisfies the requirements. This involves considering several different designs and evaluating them according to performance, cost, and ease of change. A major output of the design process is information that will guide the unit and system level testing of the software.

There has been considerable work done in both the requirements and design areas; however, a uniform and consistent set of methods has yet to be developed that covers the entire process. To date, somewhat singular efforts have been pursued that usually focus on one small step. For example, there are several software design methods that have been developed. The result is that the individual methods do not fit smoothly together, particularly at the boundaries between phases. There are two primary reasons for these problems.

First, the efforts to date have attempted to address the software issues, ignoring the distinction between software requirements and design and system requirements and design. Thus, system design remains a hardware oriented process and there is a considerably better interface between system design and hardware design.

Second, there has been a lack of thorough understanding of the requirements and design process. Recent work (LEFKOVITZ, D., 1982) suggests that if one forms a model of the work that identifies the cognitive processes that are used, then virtually all present methods have serious omissions. Further work utilizing this concept would seem to have good potential.

From a practical standpoint it is best to view the present state as a time of change. There are certainly tools and methods that can be profitably applied to the requirements and design phases; however, they do not fit well together and it is likely that new ideas and refinements will continue to emerge. It would be ideal to have methodological strings of tools to apply to the requirements and design phases; tools that assisted the engineers with the cognitive processes and the automated the recording of relevant information and generation of documentation.

A recommended approach to a SEE for requirements and design is to start with a highly integrated environment for Code and Test as previously described. Next, simplify figure 1 to reflect just the software concerns, as in figure 10. New weapon system developments would start at the top and progress through all the phases; each phase would have a support environment as shown in figure 11. It is understood that in the requirements and design phases that the tools and methods would be loosely coupled, although relationships between the phases can be determined and recorded in data bases for tracing purposes. The tools and methods are chosen off the shelf and then force fit together; several methodological strings should be implemented to gain experience with each. This approach is judged to be the most practical in the short term. As new, better integrated methods are developed they can be superimposed on this structure. The same approach can be used to extend the capabilities to the system design phases.

6. INTEGRATION FACILITIES

Integration facilities consist of a hot mockup of the weapon system computers with realistic simulation of external inputs. These facilities are used for hardware-software integration at the system level, evaluation of man-machine interfaces, and evaluation of hardware engineering change proposals. Typically, they form the hardware configuration baseline for the computer and associated subsystems. The simulation of realistic inputs allows the total system to be tested in a laboratory where sophisticated instrumentation can monitor the tests. This minimizes costly flight or shipboard testing.

Originally the integration facilities used special equipment or groups of minicomputers to simulate the external inputs; the capabilities of the test engineer were limited. Today, these facilities can take advantage of commercial computers to create an integrated test environment that both speeds the testing and takes it to greater depth. Modern integration facilities are full integrated environments and are electronically linked to the software development and maintenance computers for rapid loading of the mission software. Interactive capabilities allow symbolic debugging to be done on the target computer; extensive capabilities for storing test inputs and saving test outputs are now available.

Experience has shown that it is better to use separate computers to run the integration facilities than to attempt to use the host computer of the software production facility. This is because the Central Processor Unit (CPU) and Input/Output (I/O) utilization can be high in the integration facility computer during intense periods of real-time debugging. Further, the target computer and its subsystems frequently require extensive hardware checkout, particularly when new hardware is being developed.

7. SEE ARCHITECTURE

The goal of the SEE is to support the weapon system over the entire life-cycle. There are, of course, many ways to implement such facilities. The approach taken at the NADC was to coalesce the functions of figure 11 into two facilities as shown in figure 12. At the NADC there are large central facilities capable of supporting these activities and several integration facilities distributed throughout the Center. Therefore, one approach is to form clusters as shown in figure 13 and to interconnect the software production facilities by communications networks. Similarly, the software production facility could support just one integration facility with less capable host computers. It is important that the production facilities be interconnected regardless of size because this communications capability will ultimately permit software sharing to take place between weapon systems projects.

Alternately, the functions of figure 11 could be allocated to separate host computers that are interconnected. The choice may be dictated by the scale of the available host computers, a judgement that may vary depending on the expected workload. However, a word of caution; the software tools of today do not efficiently use computer resources, thus, it is easy to underestimate the size of the host computers. Software emulators used to unit testing take a large amount of computer resources, for example.

An emerging factor in SEE architecture is the availability of microprocessors and the expectation that networking is close at hand. An excellent example of a workstation for a software engineer is (WIRTH, N., 1981). With this technology the problem becomes how to distribute the functions to retain the dual aspects of an advanced programming system and management information system. On the one hand powerful microprocessors appear to have the power for editing, compilation, document generation, etc., but will they be capable of efficiently executing software emulators of target computers? Further, how will configuration management be enforced and how will consistent management reports be generated in such a network? A large-scale computer may still be needed to collect the software for configuration management and other management reports, as well as for executing unit testing efficiently. These problems appear to be solvable and the direction toward microprocessors seems the way of the future.

8. REFERENCES

BAUER, F.L., 1971, "Software Engineering," Proceedings of the IFIPS Congress, pp I-267, I-274.

ELIZER, P. F., May 1979, "Some Observations Concerning Existing Software Environments," DORNIER Systems, GmbH, Postfach 1360, D-7990 Friedrichshafen, Germany, Defense Advanced Research Projects Agency.

"FASP Management Summary," April 1979, U.S. Naval Air Development Center.

"FASP Software Production and Maintenance Methodology," July 1979, U.S. Naval Air Development Center.

"FASP Handbook," December 1979, U.S. Naval Air Development Center.

LEFKOVITZ, D., 1982, "The Applicability of Software Development Methodologies to Naval Embedded Computer Systems," University of Pennsylvania, Contract N62269-81-C-0455.

MORRISSEY, J.H. and WU, L.S. Y., 1980, "Software Engineering...An Economic Perspective," Proceedings 4th International Conference on Software Engineering, pp 412-422, Munich, Germany.

MUNSON, J.B. and YEH, R.T., March 1981, Report by the IEEE Software Productivity Workshop, San Diego, California.

PARISEAU, R.J., 1979, "A Screening Criterion for Delivered Source in Military Software," Report No. NADC-79163-50, U.S. Naval Air Development Center.

SOFTECH, INC., March 1979, "Support Software Planning Study," Contract N62269-74-C-0269, U.S. Naval Air Development Center.

STUEBING, H.G., "A Modern Facility for Software Production and Maintenance," AGARDograph No. 258 Guidance and Control Software, May 1980, pp 3-1, 3-14, Military Electronics Defense Expo '80 Conference Proceedings, Wiesbaen, Germany, October 1980, pp 828-845, and Proceedings of the IEEE COMPSAC '80, October 1980, pp 407-418.

STUEBING, H.G., August 1982, "A Software Engineering Environment (SEE) for Weapon System Software - Functional Description for the Code and Test Phase," Report No. NADC-82183-50, U.S. Naval Air Development Center.

WIRTH, N., March 1981, "Lillith: A Personal Computer for the Software Engineer," Proceedings of the 5th International Conference on Software Engineering, pp 2-15, San Diego, California.

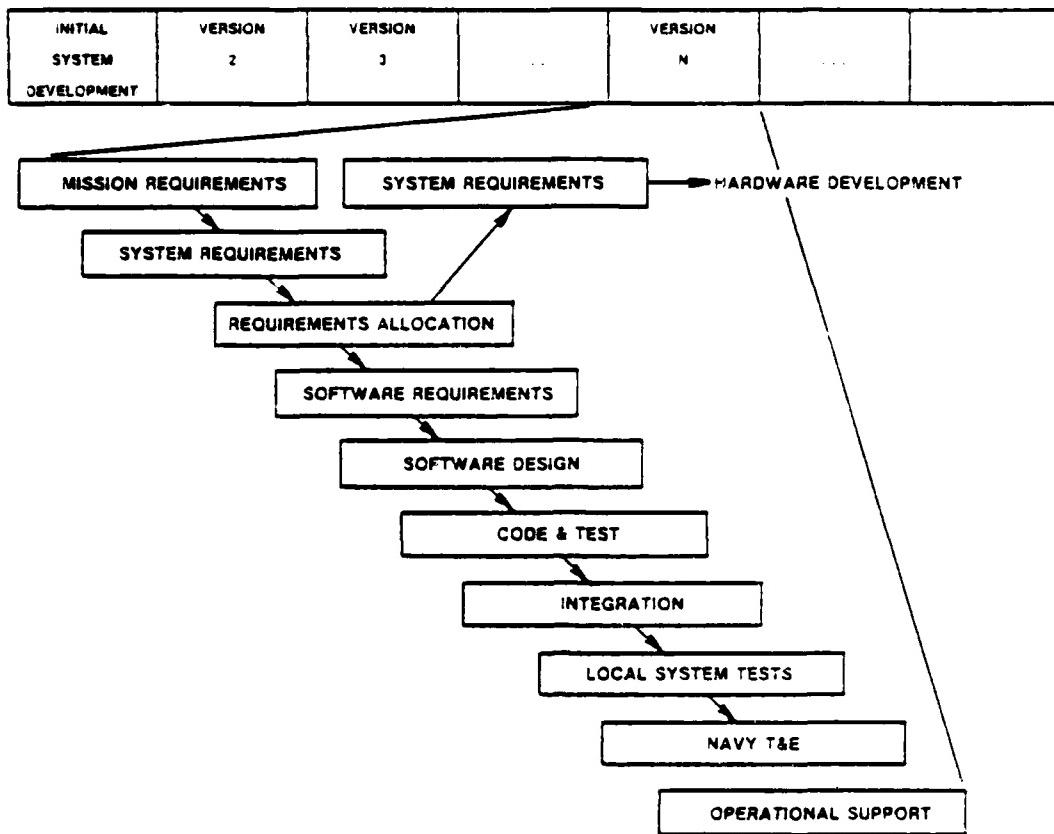


Figure 1. Generic System Development Process

FISCAL YEAR	PROJECTS	ACCOUNTS	JOBs	CPU (HOURS)	TAT (HOURS)	SOURCE LINES (MILLIONS)	OBJECT CODE (MILLIONS)
FY-76, 7T (JUL 75 - SEP 76)	3	10	57.686	197	0.53	1.6	1.6
FY-77 (OCT 76 - SEP 77)	6	20	104.652	907	2.0	2.0	2.0
FY-78 (OCT 77 - SEP 78)	13	78	110.368	1.035	2.2	2.9	2.9
FY-79 (OCT 78 - SEP 79)	28	236	105.032	1.344	0.65	3.7	3.7
FY-80 (OCT 79 - SEP 80)	35	299	118.960	1.449	0.66	7.2	8.2
FY-81 (OCT 80 - SEP 81)	41	438	135.265	1.855	0.93	12.8	12.9
FY-82 (OCT 81 - MAR 82)	47	492	67.445	1.153	0.86	14.8	16.3

Figure 2. Key Parameters Measured With The FASP

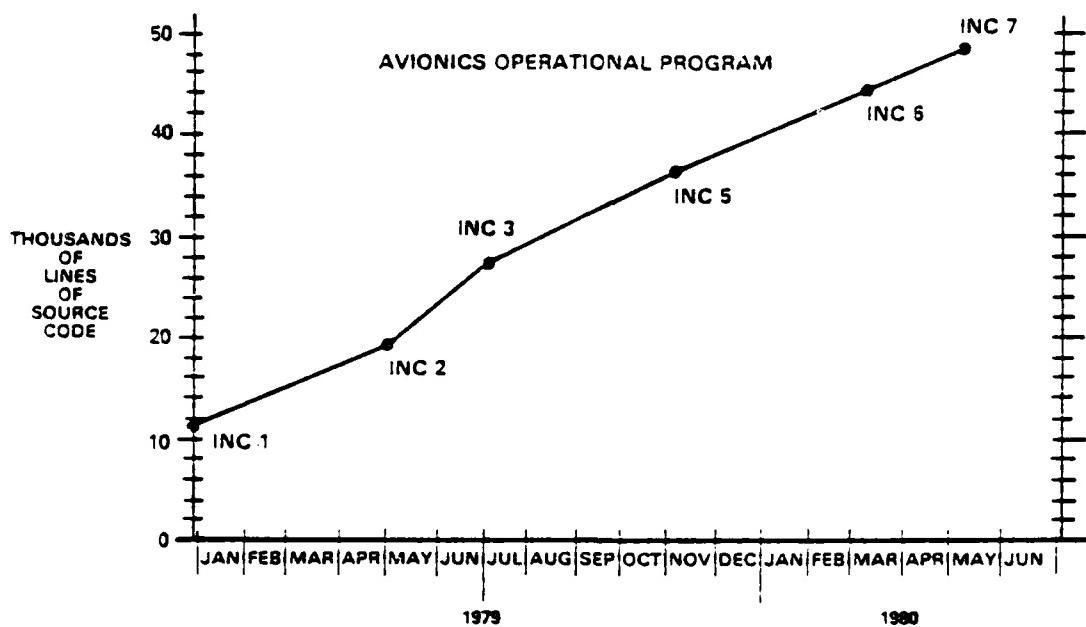


Figure 3. Delivered Source Lines

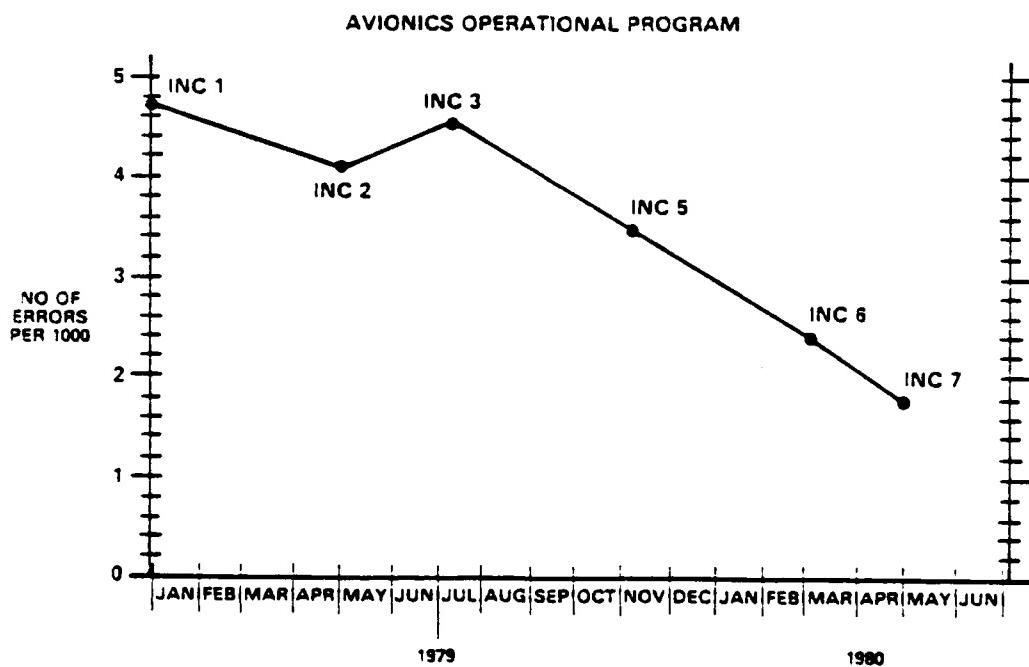


Figure 4. Number Of Software Errors Recorded

SOFTWARE DEVELOPMENT PROCEDURES
CREATE/COPY/SAVE/RESTORE A DATA BASE
DEVELOP SOFTWARE
INSTALL EXTERNAL SOFTWARE
SHARE/COPY SOFTWARE
CREATE LOAD IMAGES/TAPE
PRINT REPORTS
SOFTWARE TESTING PROCEDURES
ANALYZE SOURCE CODE
DEVELOP TESTS
INSTRUMENT CODE
EXECUTE TESTS
DEBUG TESTS
REGRESSION TEST
ANALYZE TEST RESULTS
USER ASSISTANCE PROCEDURES
LIST BULLETIN
LIST NEWS
LIST HELP
LIST USER MANUAL
CONTACT SPF PERSONNEL
SOFTWARE MANAGEMENT PROCEDURES
CONFIGURE A PROJECT
CONTROL ACCESS
PRINT PROGRESS REPORTS
IDENTIFY SOFTWARE CONFIGURATION
RELEASE SOFTWARE
TRACK STRs, SCPs, SEPs
CONFIGURATION STATUS ACCOUNTING
GENERAL PROCEDURES
DOCUMENT PRODUCTION
CONTEXT CONTROL
LOGGING ON/OFF
GLOBAL PARAMETER HANDLING
COMMAND QUEUE HANDLING
DATA CREATION
OUTPUT HANDLING

Figure 5. List Of General Procedures For Code And Test

```

Verify user access rights for this procedure
IF verification fails
THEN raise abort flag
• Notify user
ELSE
•
• (unique process flow for procedure)
•
ENDIF
IF abort flag is not raised
THEN save production data
• Make "saved" files permanent
ENDIF
Save job statistics
Make job statistics permanent

```

Figure 6. Standard Processing For Procedures

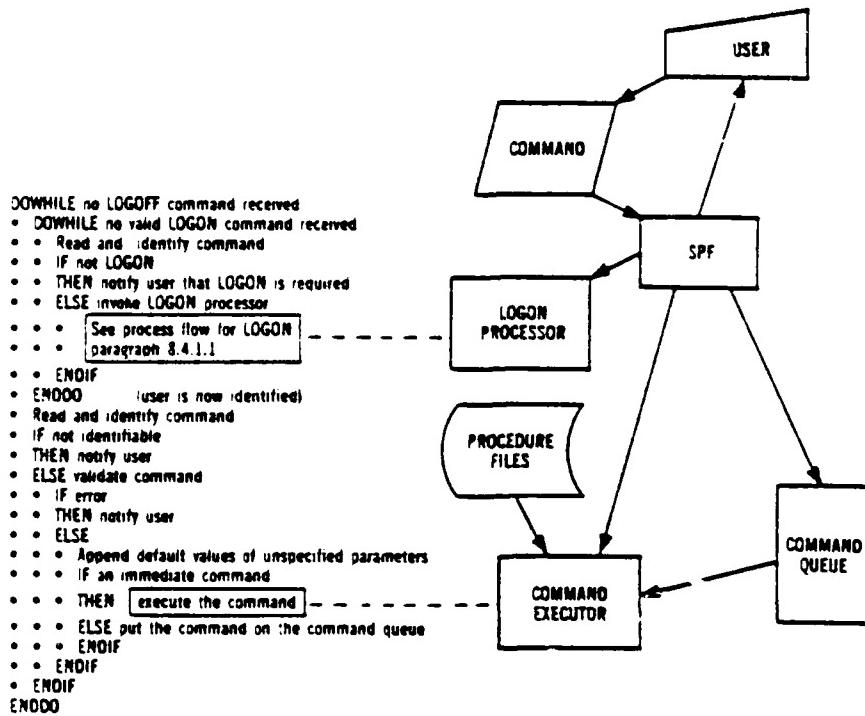


Figure 7. Command Processing Diagram

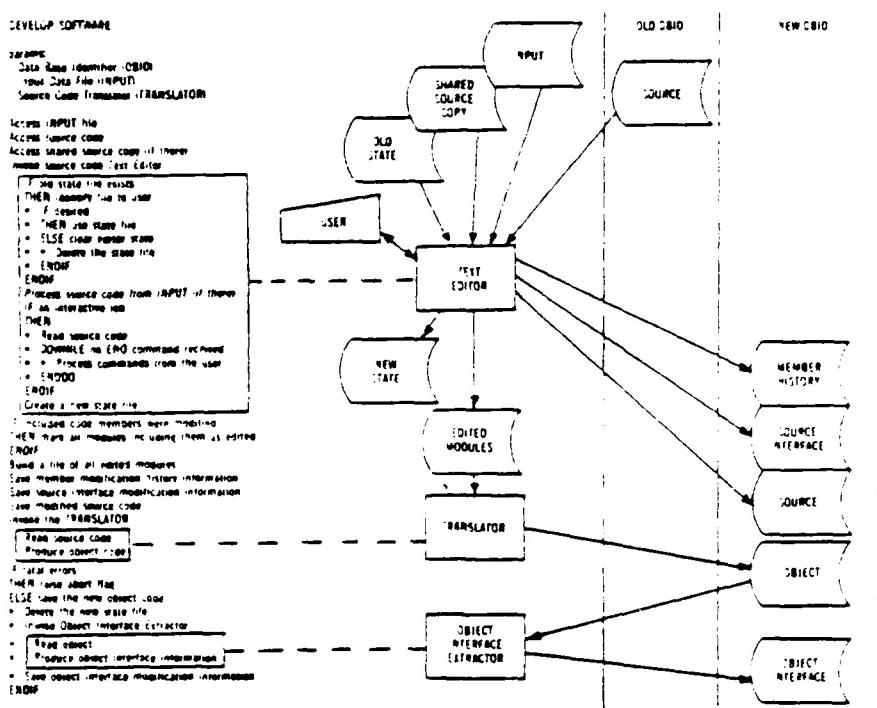


Figure 8. Process Flow Diagram

COMPUTER	AN/UY-1	AN/A-YK-14	AN/UYK-7	CYBER
COMPILER	SPL/I	CMS-2M	CMS-2Y	FORTRAN
ASSEMBLER	SPL	MACRO 20/14	ULTRA 32	COMPASS
SYSTEM GENERATOR	✓	✓	✓	✓
AUTOMATED TEST ANALYZER	✓	✓	✓	—
SOFTWARE EMULATOR	✓	✓	✓	—
MANAGEMENT REPORTS	✓	✓	✓	✓

✓ - PRESENT IN SYSTEM; — NOT PRESENT

Figure 9. Matrix Of Languages And Target Computers Of The FASP

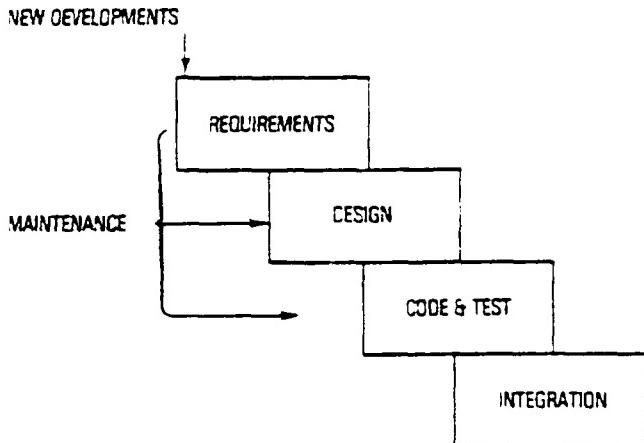


Figure 10. Simplified Development Process

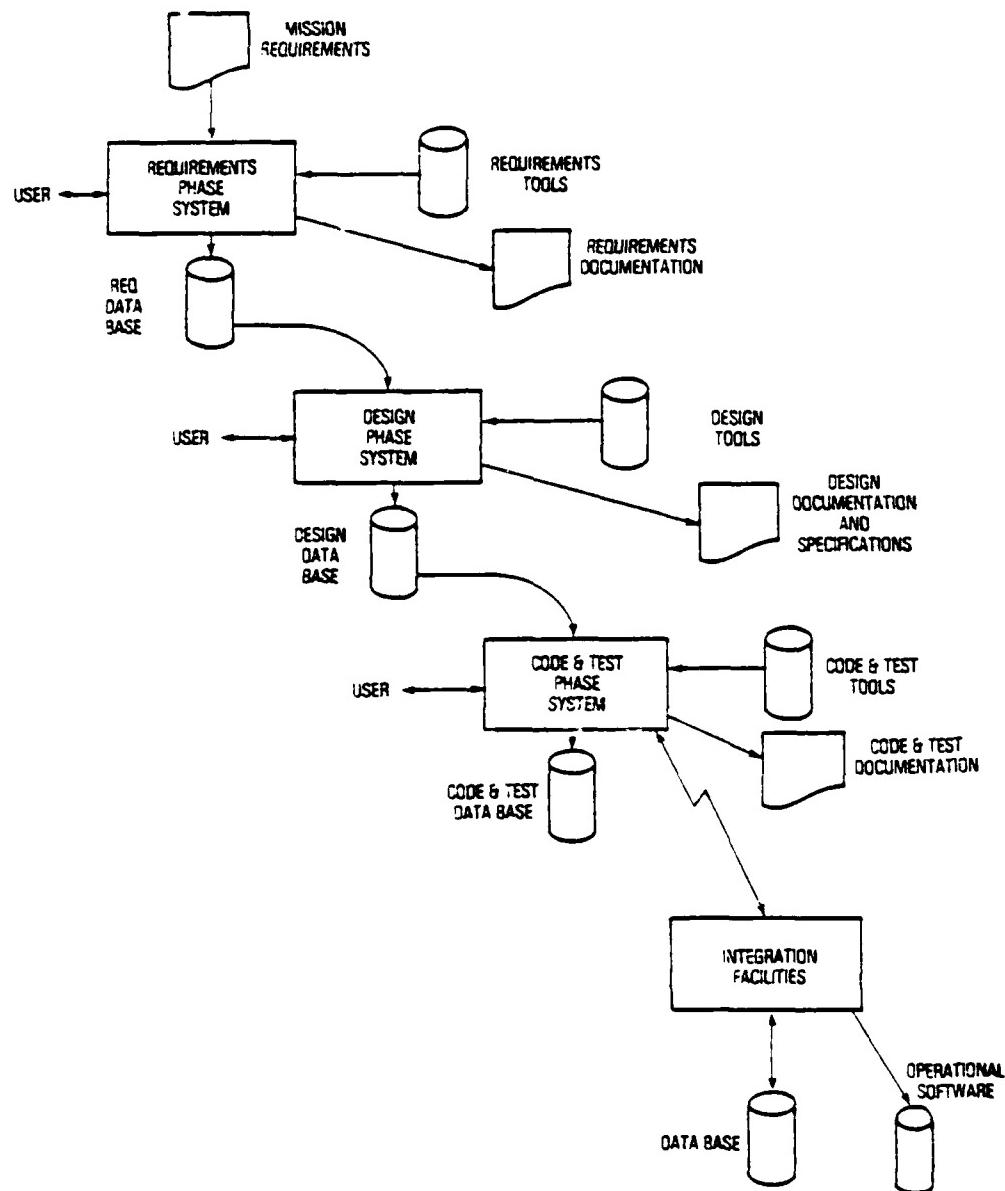


Figure 11. Separate Facilities For Each Phase

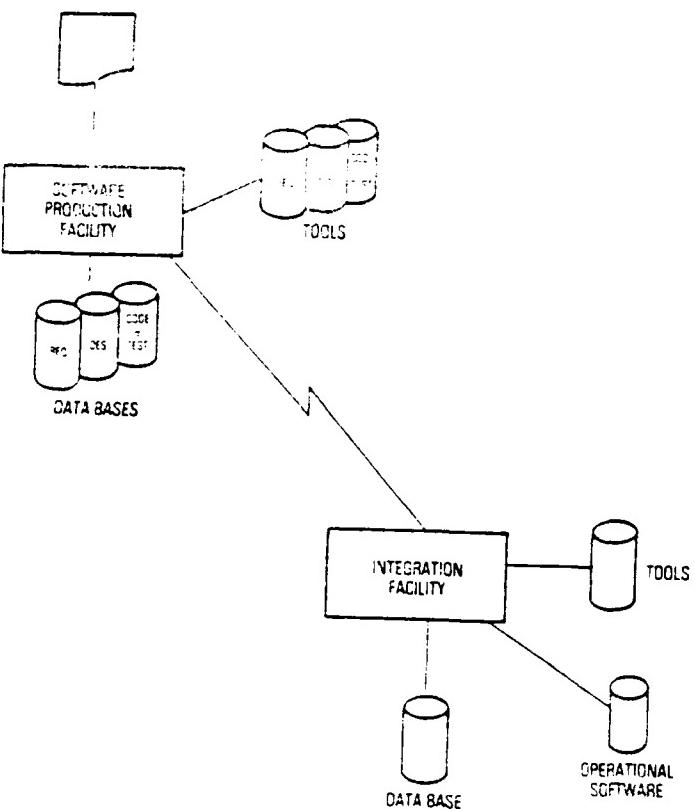


Figure 12. The NADC Two Facilities System

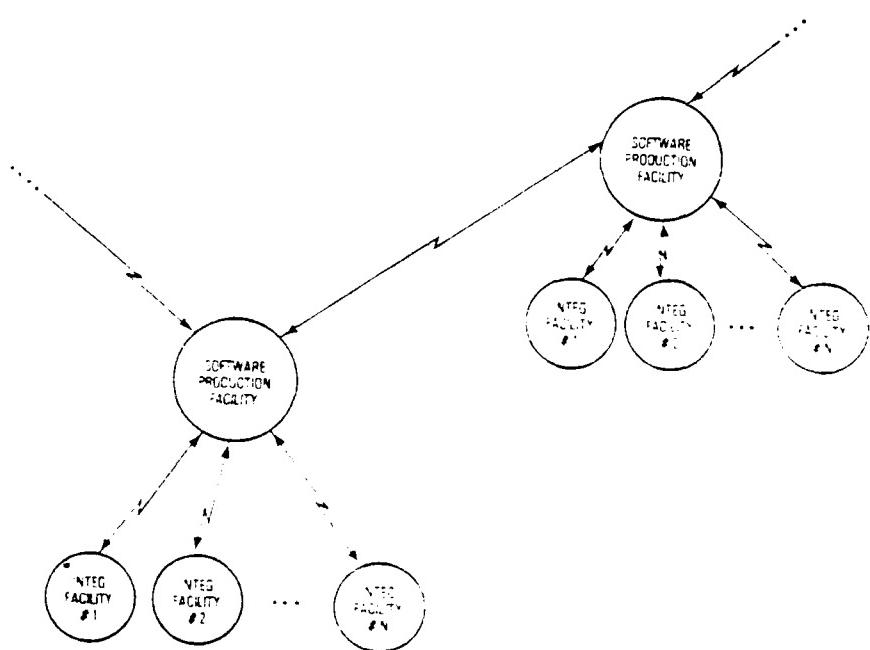


Figure 13. Interconnection Of Software Production Facilities

ORLANDO I

FINAL REPORT

PANEL E

THE SOFTWARE CHANGE PROCESS

CO-CHAIRMAN: Mr. J. (Joe) Black
WR-AFLC/MMRR
Robins AFB, GA 31098
(912) 926-5948
A/V 468-5948

CO-CHAIRMAN: Mr. J. (Jack) Cooper
CACI Inc., Federal Penthouse
1700 N. Moore St.
Arlington, VA 22209
(703) 276-2826

TABLE OF CONTENTS

	<u>Page</u>
4.5 PANEL E The Software Change Process	
4.5.1 Scope	4-5-1
4.5.2 References	4-5-2
4.5.3 Definitions	4-5-2
4.5.4 General Policy	4-5-2
4.5.4.1 Management	4-5-6
4.5.4.2 Configuration Management	4-5-6
4.5.4.3 Change Engineering	4-5-6
4.5.4.4 Testing	4-5-6
4.5.4.5 Software Quality Assessment and Management (SQAM)	4-5-8
4.5.5 Change Requirements	4-5-8
4.5.5.1 Introduction	4-5-8
4.5.5.2 Requirements for Change Definition	4-5-9
4.5.5.3 Management of Change Definition	4-5-10
4.5.5.4 Management During The Change Process	4-5-11
4.5.6 General Management Controls	4-5-12
4.5.6.1 System Manager	4-5-12
4.5.6.2 Software Support Activity	4-5-12
4.5.6.3 Software Manager	4-5-12
4.5.6.4 Use/User Representative	4-5-12
4.5.6.5 Internal Interfaces	4-5-13
4.5.6.6 External Interfaces	4-5-13
4.5.6.7 Task Identification and Control	4-5-13
4.5.6.8 Change Requirements Review	4-5-14
4.5.6.9 Systems Requirements Review	4-5-14
4.5.6.10 Top Level Design Review	4-5-15
4.5.6.11 Test Plan Review	4-5-15
4.5.6.12 Detailed Design Review	4-5-16
4.5.6.13 Test Procedures Review	4-5-16
4.5.6.14 Test Report Reviews	4-5-17
4.5.6.15 Independent Test Readiness Review	4-5-17
4.5.6.16 User Test Readiness Review	4-5-18
4.5.6.17 Acceptance Review	4-5-18
4.5.6.18 Status Review	4-5-18
4.5.6.19 Priority Change Process	4-5-19
4.5.7 Technical Controls	4-5-20
4.5.7.1 Scope	4-5-20
4.5.7.2 Software Engineering Change Process	4-5-20
4.5.7.3 Test Function	4-5-21
4.5.7.4 Software	4-5-25
4.5.7.5 External Interface	4-5-28
4.5.8 Pre-Deployment Responsibilities of the Software Support Activity (SSA)	4-5-29
4.5.8.1 General	4-5-29
4.5.8.2 Concept Phase	4-5-29

TABLE OF CONTENTS (Cont)

	<u>Page</u>
4.5.8.3 Design and Development Phase	4-5-30
4.5.8.4 Acceptance Phase Activities	4-5-31
Appendix - Panel E Participants	A-1

4.5.1 SCOPE

The purpose of this publication is to provide a uniform policy structure within which all Department of Defense support agencies can function to provide efficient and timely software change support for deployed weapon systems. The guidelines herein are intended to cover all categories of operational equipment and attendant and/or embedded software that are furnished to using commands and supported by DOD logistics support agencies. It is understood that any policy manual providing guidance for equipment as diverse as submarines, tanks, and fighter aircraft must address the most generic of the required policy areas and allow reasonable flexibility in the delineation of specific policy statements intended to cover various DOD organizational structures and diverse equipment requirements. Though not specifically directed as a part of this policy, it is highly recommended that every department of defense agency review the entire scope of software support required with a view toward structuring equipment and weapon systems support taxonomies which provide reasonable economies of scale, standardization and facility utilization. Ensuring a chain of command reporting sequence that provides the software support agency an adequate voice in the control of resources control and allocation methodologies is mandatory.

This manual has been derived by applying analytical techniques intended to subdivide the software change process into generic segments without regard to organizational make-up or functional allocation within any particular service. Multiple services and agencies have contributed lessons learned and experiences gained. The techniques utilized in supporting deployed weapon systems where significant capability is derived from the software embodied therein represent the experiences gained to date and reflect only a minor subset of those anticipated to be experienced within the coming decade. As such, it is intended that this manual be reviewed and updated on an annual basis to keep pace with the shifting support requirements generated by the increasing knowledge base of the using and supporting commands and the on-going technological innovations that continue to materialize from the industrial base which supports Department of Defense activities.

The beginning point for all policy generated herein has been a generic software change implementation model which subdivides the change process into its fundamental discipline based requirement areas. These are management controls, configuration management, software engineering, software quality assessment, and management and technical controls. In addition, the key interface areas for requirements derivation as a beginning point and user acceptance as a configuration stabilization point have been addressed. To apply this manual appropriately, it is necessary to understand that software maintenance is a term brought about by usage which distorts the understanding of the software change process itself. From an overall viewpoint, the software change model appears to be a development cycle with different acronyms and descriptions of the software development life cycle. In terms of management controls, facility requirements (including hardware and software support environments) and the operating environment context, this analogy leads to a poor understanding of the extensive requirements of an efficient post deployment support capability. Unlike the well structured requirements process which governs any major systems acquisition where significant software is involved, the software support activity is faced with the delayed accumulation of changed requirements which exceed the resources base available. These facts make necessary a very active management control process with significant interface to the organizations who generated the change requirements. In addition, a significant degree of real time assessment by the management control structure, including the using organization, is

required to exercise decision processes geared to include or exclude specific change request during the active portion of any on-going software change cycle.

4.5.2 REFERENCES (to be developed)

4.5.3 DEFINITIONS (to be developed)

4.5.4 GENERAL POLICY

Planning for accomplishing the software change process during the post deployment phase will commence during the Concept Definition phase of the system life cycle as defined by DODI 5000.1. This planning will identify all internal and external interfaces, the responsible software support organization and the estimated resources (manpower, facilities, support environment and documentation) required to adequately perform PDSS. To insure that the change process is implemented in a complete, timely, cost effective, and orderly manner, each service will create a standard change process following the DOD generic software change model described in this section. The five major functional areas will be addressed as identified along with critical external interfaces. The service level implementation of this policy contained in Appendices A through D shall identify the change policy for each category of software, e.g., operational, support, trainer, etc., and their interrelationships.

Figures 4.5-1 through 4.5-4 are constructed as follows: blocks represent discrete functions/activities of the change process; lines indicate a product which could be a decision, a document, or a product. Figures are not scaled with respect to time; however, functions are in time sequence. The figures identified herein are generic in nature and serve as an implementation guide for each element of DOD in structuring internal policy (see appendices).

Figure 4.5-1 identifies the top level activities required in the generic software change process. It separates in a simplified manner the chief functional purpose of each activity to one of five major functional areas. Complex interrelationships are required between the various functional areas as highlighted indicating the necessity for adequate pre-planning implemented by reasonable and limited organizational divisions within the overall software change process.

No intent is expressed as to correct or accept organizational methods since considerable tailoring is necessary to allow optimum processes for the various DOD components and/or equipment taxonomies.

Although the MIL-STD-SDS was prepared for and organized around original development activities, the ability to correlate specific activities and the progressive baseline formulation process is equally valid during the follow-on support life cycle. Figure 4.5-2 serves to provide correlation between the software process change model activities and the MIL-STD-SDS documentation products and reviews. The iterative nature of the PDSS activities serves to reinforce the requirements for usable valid software baseline documentation.

Figure 4.5-3 is the same as Figure 4.5-2 with additional emphasis on the software change products represented as data deliverables time phased with the software development process specified in MIL-STD-SDS. Specific products should be developed and delivered with a continuing software change process envisioned. All documentation should be delivered in a format and on a media providing maximum

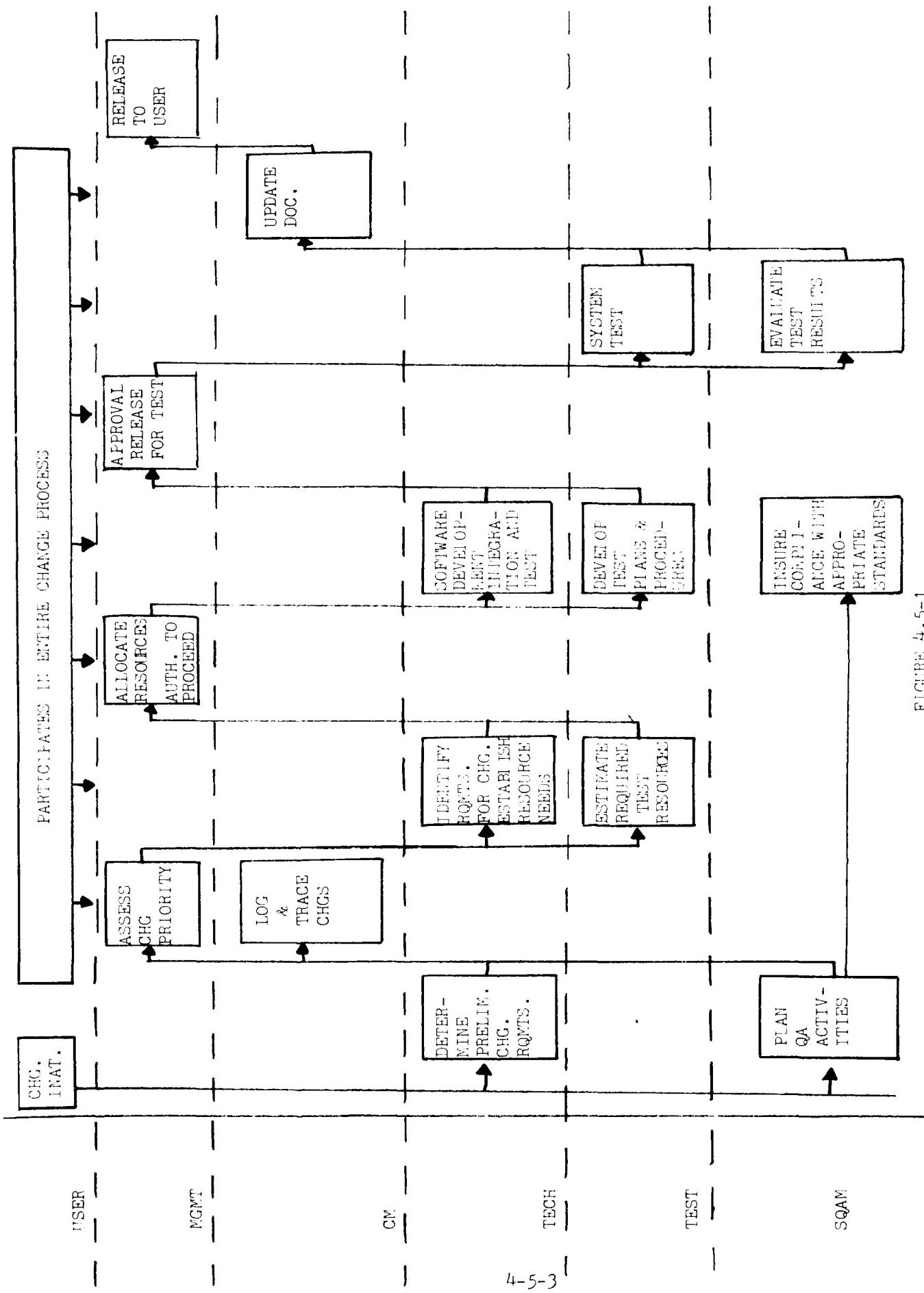
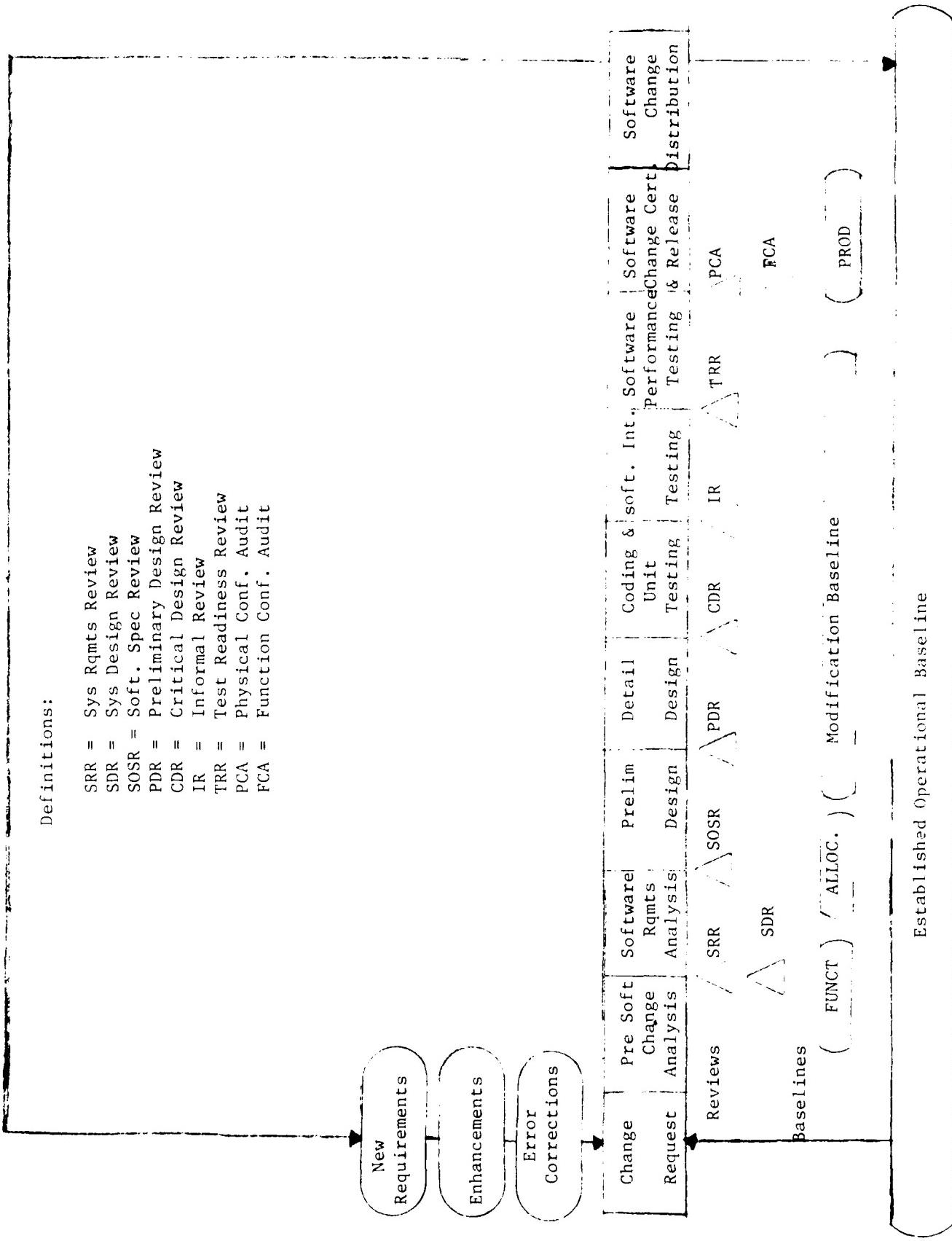


FIGURE 4.5-1



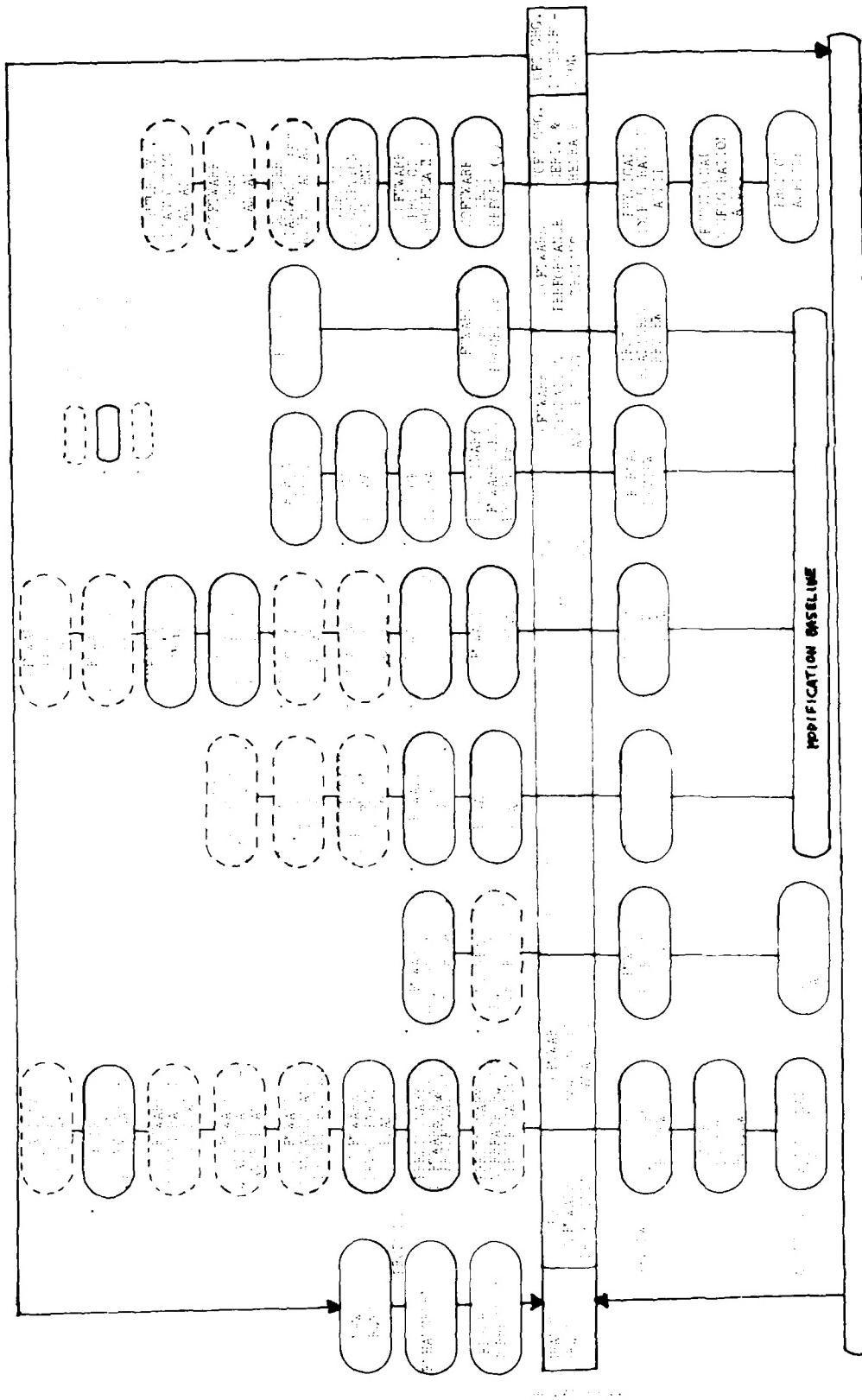


Figure 4-3

long-term utility to the PDSS activity.

Figure 4.5-4 is a detailed generic flow of the software change process in the post deployment phase. This detailed flow model is provided to serve as a guide in developing service specific software change processes. The model in conjunction with this policy outline is intended to guide those agencies and organizations initially establishing or assessing the adequacy of its software support infrastructure.

Figure 4.5-4 is intended as a very detailed identification of functions and sub-functions. It serves as a guide for each service in their facility planning and implementation and as a "checklist" in the conduct of the individual change process. The formal detailed change process functions and controls shall be commensurate with the complexity of the change.

Implementation of and operational use of the model is dependent upon: (1) clearly established interfaces with the product user for change request definition, priority, and urgency, and (2) interface with the acquisition community for Planned Product Improvements or technology insertion which require support system modification/upgrade/update.

Implementation of the model interfaces with logistical support in the acquisition/development of facilities (brick and mortar) and equipment and tools, and in the acquisition and training of human resources. Other interfaces exist with: (1) operational field/fleet/flight test ranges or facilities, (2) publication and distribution of products and associated documentation, and (3) training of operational support personnel necessary to implement changed products and associated documentation.

4.5.4.1 MANAGEMENT

A close and continuous interface with the product user shall be established and maintained. The interface shall be institutionalized in mission statements of pertinent organizations.

Adequate planning shall be performed and controls emplaced to insure sufficient resources are available, program visibility is maintained, schedules and costs are monitored, and all administrative functions are performed. Specific tasking shall be levied to all participating organizations.

4.5.4.2 CONFIGURATION MANAGEMENT

Configuration management function shall be performed throughout the change process to adequately control all the baselines during the change process.

4.5.4.3 CHANGE ENGINEERING

The software development portion of the change engineering process shall include the techniques and control methods specified in MIL-STD-SDS.

4.5.4.4 TESTING

A test philosophy commensurate with the software type and the criticality, complexity, and scope of the change forecast shall be developed during initial

facility development. Resources shall be programmed to support the implementation of the test program during the change process. Requirements for contractor and organic software test beds must be addressed early to minimize duplication.

Test requirements shall be coordinated with the user community. Test requirements shall be generated concurrently with change requirements. Need for IV&V shall be determined during change requirements evaluation.

Test plans and procedures shall be generated as required throughout the software change cycle for each software change cycle.

4.5.4.5 SOFTWARE QUALITY ASSESSMENT AND MEASUREMENT (SQAM)

A SQAM program shall be planned during the initial change facility development and implemented throughout the change process.

4.5.5 CHANGE REQUIREMENTS

4.5.5.1 INTRODUCTION

PDSS is the performance of those activities required to keep a software system or software controlled system operational and responsive after it is accepted and placed into operation. PDSS, then, refers to the set of activities which result in changes to the originally accepted (baseline) product set. These changes consist of modifications created by correcting, inserting, deleting, extending, and enhancing the baseline system. Generally, these changes are made in order to keep the system functioning in an evolving, expanding user and operational environment.

4.5.5.1.1 Source of Changes

Changes to existing software can be initiated by the designated user representative, the software support activity or by external influences. Regardless of the source, maximum coordination between affected organizations is required in order to maintain system effectiveness, and to ensure user acceptability. The decision to alter a systems' software shall remain with the organization that exercises operational control over the system.

4.5.5.1.2 Categories of Change

Software changes can be divided into four categories:

a. Corrective Changes - Corrective changes are those changes made to correct errors in the current software baseline. These errors may be inherent in the software design logic or in the actual coding.

b. Enhancements - Enhancements are changes which are made to improve software performance, maintainability or understandability. Enhancements are generally made to augment or fine tune the software. Optimization of code to make it run faster or take less memory is an example.

c. Adaptive Changes - Adaptive changes refer to modifications made to satisfy or accommodate changes in the environment in which a software system must operate or to add a capability not originally envisioned or required in the current baseline. Adaptive changes enhance the useful life of the software and the overall

system. Modifications of the software add interfaces with other systems or add new logic features to accommodate an operational change.

d. Emergency/Urgent Changes - Emergency and urgent changes will interrupt any on-going software changes and will be incorporated into the baseline that is currently being used. The priority of the change will dictate the expediency placed on all points during the software change process.

4.5.5.2 REQUIREMENTS FOR CHANGE DEFINITION

A system for reporting proposed software changes of any of the three types discussed in Section 4.5.5.1.2 shall be established and documented in the Computer Resources Life Cycle Management Plan (CRLCMP). The reporting system shall include the technical, scheduling and management considerations listed below.

4.5.5.2.1 Technical

Requirements must define the required operational capabilities in terms of inputs, processing, outputs, accuracy and load. A detailed evaluation of the software requirements can play a major role in ensuring that they possess characteristics that contribute to a quality product. Requirements validation may require a wide range of activities, including the conduct of extensive traceability analysis and the development or use of detailed models and analytical simulation to evaluate both functional and performance aspects of requirements. Criteria shall be established to determine emergency requirements.

4.5.5.2.2 Schedule

The responsible organizations involved shall develop a schedule that is mutually agreed upon which consolidates routine changes, maximizes resource utilization, meets all organizations' needs, and allows for adequate training. Emergency changes shall take precedence over routine changes.

The final decision in scheduling requirements implementation must, however, be that of the activity which exercises operational control over the system.

Because of the ever increasing complexity of interoperability requirements, both intra- and inter-service, it is essential that software modifications be scheduled in such a manner as to preclude disruption of established capabilities. In this regard, it may be necessary to postpone implementation of a validated requirement in one system until a compatible requirement(s) can be inserted into others. Another important criteria to be considered in scheduling changes is that of training. Except for those changes that are transparent to the operator, all scheduled releases must coincide with the ability of the responsible training activities to retrain users, and to provide the necessary modifications to operator and training manuals.

4.5.5.2.3 Miscellaneous

In some cases failure to implement a software modification can result in a critical operational deficiency which may adversely affect the effectiveness of the total system. In these cases, it shall be necessary to document the impact these situations will have on the affected operations. Specifying and maintaining

established standards shall be a major consideration in requirements generation activities. Adherence to applicable DOD software standards will significantly enhance operational and interoperability capabilities.

4.5.5.3 MANAGEMENT OF CHANGE DEFINITION

4.5.5.3.1 General

Each requirements generation activity shall develop and maintain management procedures which defines the controls, coordination and documentation necessary for PDSS. This section discusses the policy and procedures for the change requirements specified in Section 4.5.5.2.

4.5.5.3.1.1 Management Controls

The change requirement management procedures shall specify the controls needed to effectively evaluate, assign, and prioritize the software change requests. The procedures shall also define controls necessary for approving/rejecting a software change. Change requirements shall adhere to established standards and performance criteria.

4.5.5.3.1.2 Coordination

The procedures shall specify the organizational activities and procedures for coordination of proposed modifications. As a minimum, all activities which operate or maintain interfacing systems must be included in the plan.

4.5.5.3.1.3 Documentation

The policy shall specify the type and content of the documentation necessitated by software change, and enforce existing documentation standards and procedures.

4.5.5.3.2 Organizational Responsibilities

The organization responsible for the management of change requirements has the responsibility to keep all systems functioning and responsive to user requests. This includes the establishment of standards, guidelines and procedures which describe in broad terms the responsibilities, authorities, functions, and operations of the SSA, USER and any other external organization involved with or affected by a change to the software.

In order to maintain responsiveness, it is essential that the establishment of procedures for the submission, review, approval or rejection of change requests be developed. These procedures shall delineate the specific authorities of the responsible organizations.

This shall include, but not be limited to procedures for designating the responsibility for:

- Preparation and submission of formal change requests.
- Limiting changes processed to only those approved.
- Evaluation of type and frequency of change.

Organizational responsibilities shall also include specification of

procedures by which all of the groups involved with or affected by a software change must coordinate.

4.5.5.3.3 Resources

Proper allocation of resources to the activity generation requirements will significantly enhance the software change process. The organization generating the requirements shall have identified procedures for requesting the proper resources, such as tools, personnel and facilities, to accurately define system and system software requirements. Detailed simulations and computer driven models should be considered as a necessary expedient in pre- and post-deployment phases and should be made available to the requirements generation activity as early in the cycle as possible where justification for their acquisition is adequate.

4.5.5.3.4 Schedule

During the process of defining the change requirements, the anticipated or required schedule should be developed. This schedule should address significant milestones in the change process and significant external events such as required dates for interoperability with other systems. The schedule should also address the combination of proposed changes in block cycle format to ease the duplicative management functions, if the priority of the proposed changes permit.

4.5.5.3.5 Acceptance Criteria

After joint agreement regarding the requirements which will be addressed by the software modification, the organizations shall determine the criteria for acceptance. At a minimum, the criteria shall include performance criteria and the methods and agencies responsible for verification that the requirements are achieved.

4.5.5.3.6 Interface Procedures

Interface procedures shall be established to ensure that all affected organizations are included in the change definition review process so that the total impact of the change is known when the final change determination is made. Affected organizations shall commence actions necessary to complete required changes on schedule.

4.5.5.4 MANAGEMENT DURING THE CHANGE PROCESS

4.5.5.4.1 General

This subject is discussed in greater detail in Section 4.5.6 of this manual. This section addresses the planning required during the requirements change definition process needed to ensure that management during the actual change process will progress smoothly.

4.5.5.4.2 Management Controls

The organization making the change shall ensure that a system of periodic reviews and audits is established to manage the change process. An internal baseline management system shall be established to ensure adequate tracking of the change requirements through the change process.

4.5.5.4.3 Release Acceptance Criteria

Planning for acceptance criteria of the changed software shall begin at the time the requirement for the change is identified. Specific elements such as schedule, operational and functional capabilities, required testing, and training packages shall be included in the acceptance criteria. A mechanism for certification of the changed software to ensure that the change requirement has been adequately resolved and procedures for final sign-off acceptance shall be developed.

4.5.6 GENERAL MANAGEMENT CONTROLS

This section focuses on the management controls which, of necessity, overlay the entire software change process. In order to gain the required focus, the software change process and the software manager are addressed as distinct and separable entities.

This is recognized as being a simplifying assumption. Hardware and software are integrated systems and are typically inseparable from a management and engineering control standpoint. Moreover, in any bureaucratic system there are multiple levels of management, each retaining some level of authority and control. Control is, therefore, addressed functionally with overriding emphasis on the system management function and the software management functions, irrespective of the number of indentures characterizing any given Service's implementation. Each service shall insure procedures and controls are adequate to provide administrative control in the host organizational structure.

4.5.6.1 SYSTEM MANAGER

Within any generalized support structure there must be a system manager to serve as focal point for all change activity on the system. As herein used, a system defined as any major defense system detailed in DODD 5000.1 and 5000.2. All changes (hardware or software) initiated will be directed to this system manager, and he shall accomplish appropriate actions to acquire and control necessary resources to accomplish updates and changes to the system.

4.5.6.2 SOFTWARE SUPPORT ACTIVITY

Based on varying levels of system complexity and integration, there may exist one or more software support activities which retain subsystem level support responsibilities for individual weapon systems. Organizationally, these activities may be placed with a variety of different government service or contractor agencies; however, to the extent that the software support activity is engaged in a change to a given weapon system, it is ultimately responsible directly to the system manager.

4.5.6.3 SOFTWARE MANAGER

Within the given software support activity there must exist a software manager responsible for control of the change process. This individual is delegated certain responsibilities and authorities by the system manager and within that framework is ultimately responsible to produce and deliver changes as tasked and approved by the System Manager.

4.5.6.4 USER/USER REPRESENTATIVE

The user is defined as that individual or organization whose operational mission establishes the requirements which the software is designed to satisfy. In this context, the user is a critical element in defining and prioritizing changes to the requirements baseline. The user, or user representative, must interface closely with both the system manager and the software manager throughout the system life cycle to coordinate the software change process.

4.5.6.5 INTERNAL INTERFACES

By its nature, the management function must interface with each of the other major functions involved in the software change process. The majority of the internal interfaces are accomplished by the software manager as integral parts of the change management and resource control function.

4.5.6.6 EXTERNAL INTERFACES

To be effective the software change process must blend together the technical resources controlled by the software manager, with financial resources controlled by the system manager and operational requirements controlled by the user.

These represent the two major external interfaces for the software support activity. These two external activities in turn provide the linkages required by the software manager to indirectly interface with the operational environment and the resource allocation and control environment.

Additional external interfaces shall be directed through the system manager. As a minimum all external interfaces shall receive copies of the proposed updates to baselines for which there is an interface impact.

4.5.6.7 TASK IDENTIFICATION AND CONTROL

All change requests shall be directed to the system manager. In turn the system manager passes the formal change request to the software manager of the software support activity and jointly they assign a control number. The software manager then distributes copies of the change request to configuration management, engineering, test and quality control to generate the preliminary change requirements and estimate resources needed. An assessment of the change priority shall also be conducted by the system manager, the software manager of the software support activity and the user of the applicable system.

Inputs from the originator of the change include copies of the change request which are distributed to the disciplines of the software support activity and a ranking assigned based upon other changes. After the accomplishment of task assignment has passed from the system manager to the software manager, it is passed to the engineering and test organizations. Normal task controls are then passed to the next management control point, the Change Requirements Review.

The system manager passes on the change request to the software manager of the software support authority. They work together to assign a task control number for tracking purposes and with the user of the applicable system to assess the priority of the change in relation to other identified system changes. The software manager then passes the change request and task assignments to the other disciplines in the software support activity.

As the engineering organization begins processing and clarifying the change requirements, they complete initial classification of the type of modification to the system that would be most mission effective. If this evaluation determines that the most effective modification would be to the hardware, the initial evaluation is fed back through the software manager to the system manager for further task assignment.

4.5.6.8 CHANGE REQUIREMENTS REVIEW

Prior to directing a software change, management must insure that the requirements are baselined and that an adequate estimate of resources and schedule have been accomplished.

After a task is received, it must be analyzed to insure that it is completely stated and understood. An in-depth study is made of all proposed software changes to determine the system elements affected as well as the different systems which the change affects. These changes and the resources necessary to accomplish them are provided to the requirements review.

All requirements are stated in a user friendly language and in enough detail to insure that the exact same requirement is understood by both user and software personnel. All requirements must be stated completely to include any interoperability impact so that all aspects of change can be prioritized for assignment of resources. Estimates of man/machine requirements as well as time constraints need to be expressed. If a limitation of resources prevents all requirements from being accomplished then it is imperative that a priority list of changes or other possible options be provided.

It is the responsibility of the software manager to ensure that all requirements are properly analyzed, reviewed, and prioritized. This is accomplished by having all concerned agencies review and approve the requirements list prior to management deciding to commit the necessary software effort.

After the engineering analysis and the requirements review, the estimated cost and schedule, along with any options are provided to the system manager. The system manager has to determine if adequate resources may be expended to accomplish the primary request or other option.

4.5.6.9 SYSTEMS REQUIREMENTS REVIEW

The systems manager will review the degree and manner in which his system will be altered or affected by the proposed software change. A refined cost and schedule estimate must be identified for the proposed software change. The software manager must ensure that sufficient resources will be available, including any peculiar subsystem components to completely execute the change process.

The system manager will receive a report from the software manager identifying the proposed software change and the change in functional system performance expected from implementation of the change. The system manager and the user representative will jointly review the change request and will either approve or disapprove the change. In cases where the system manager and the user representative disagree, the decision to change the software shall be made by the user's representative. Once approved, the system manager will attempt to secure

appropriate funding for the software manager to implement the change.

Once a software change has been approved, the system manager is responsible for providing funding and any required system components to the software manager. He is also responsible for ensuring the coordination of the software change with all other sub-system managers.

The engineering, test and CM resource requirements will be integrated and converted to costs by the software manager. The software manager will provide the final and official position for costs and schedules for the proposed change to the system manager for approval. The system manager will provide required funding and will maintain a system schedule that includes all major system changes. He will distribute this schedule to the software manager and user.

4.5.6.10 TOP LEVEL DESIGN REVIEW

The top level design baseline package will be reviewed by all concerned agencies to determine its acceptability. The software manager will coordinate inputs from the engineering manager and the QA manager.

The software manager will use the system manager's approved cost and schedule data with the engineering manager's performance package to establish the top level design baseline. The output of this effort will be either an approved or rejected top level package by the software manager. This package will also be provided, if approved, to the CM manager.

The software manager is responsible for implementing approved changes to the software baseline. This action sets the stage for the development of a proposed detail design. The software manager must direct commencement of detail design to the engineering, test and CM managers. In addition, an information package should be provided to the system manager and the user representative. A final review of the impact the approved change will have on the system level schedule and the adequacy of resources must be made by the software manager. Resource or scheduling problems must be reported to the system manager and user representative for resolution.

4.5.6.11 TEST PLAN REVIEW. Each change of the software system must be tested to insure that the system has not been degraded and that the system performs to specification as well as for interoperability impacts. This is provided for in the test plan.

Each new requirement must be tested. To insure that all requirements are tested, there must be a trace from the requirements list as well as any changes to the top level design. The test plan must be reviewed and approved before effort is continued on preparation of test procedures. The review will include representation from all systems that have identified as having interoperability impacts.

The test plan should provide the overall concept of the test. It should have a detailed list of personnel and hardware requirements and other unique requirements should be identified. All software changes being made should be clearly traceable. The test plan should provide input for preparation of detailed test procedures, and it provides the manager with planning time to request any additional assets required for the test.

It is the responsibility of the software manager to insure that all changes to the systems are adequately tested prior to release for independent testing. A test director is normally appointed and given the authority to access whatever activity that is necessary for the successful conduct of the test.

Any inputs affecting the cost and schedule that were not previously identified should be stated.

4.5.6.12 DETAILED DESIGN REVIEW

A review of the detailed software design by all concerned agencies to determine suitability to commence coding of the software change. This action will lead to the initiation of user manual updates.

The data and information provided to the software manager for decision making will include an acceptable test plan and a performance package identifying the detail design. The output of this review, if approved, will be an approved detail design baseline.

The software manager has the authority to approve or reject the performance package which establishes the detail design baseline. He is responsible for providing authorization for the CM manager to commence with a detail design baseline change and initiation of user manual update. He is also responsible for authorizing engineering manager to commence generation of code reflecting the software change. The software manager must inform copy approval of the change to system manager and the user community.

If the design is approved the software manager does not need to take action in terms of cost and schedule unless changes are expected at this time. If the design is rejected a proposed revised cost estimate needs to be developed. This must be transmitted to the system manager and user community.

4.5.6.13 TEST PROCEDURES REVIEW

The detailed test procedures must be reviewed to insure that every requirement change being implemented is traced to a test procedure. The test procedures should provide for a degree of testing to insure that the overall system has not been degraded. The procedures must be reviewed for completeness and accuracy. The procedures should also be reviewed to insure that parametric testing is accomplished, that incorrect entries are also tested and maintainability/supportability has been evaluated.

Personnel preparing test procedures must be a part of the design process and have access to all meetings and documentation which dictate how the software is to be written. The requirements list and top level design changes must be written in a testable manner if test procedures are to be meaningful. Test procedures should be reviewed to insure that the sequence can be followed during the test. Being able to repeat the sequence allows for a problem to be reproduced as necessary to provide additional information for debug efforts.

It is the responsibility of the software manager to insure that the test procedures are validated prior to the system test. The test director should be provided adequate personnel, equipment, and time to test all procedures. All identified deficiencies must be corrected or dispositioned by the software manager

prior to acceptance of the subject test plan.

Adequate personnel and time should be planned to provide for any additional retesting of procedures caused not only by software but also by hardware failure or operator error. If additional resources are needed, the software manager shall get approval from the system manager prior to the start of the test phase.

4.5.6.14 TEST REPORT REVIEWS. There are three levels of test reports that shall be generated as part of every software change process to aid in the tracking of the test program. The first report is generated after the internal module testing and integration is completed. The second report is generated after the independent system performance testing is completed. The final test report is generated after the user acceptance testing is completed.

During the review of each test report all test procedure results are analyzed to determine validity and completeness. In addition, recommended changes to the user manuals are also reviewed.

In order to conduct test report reviews a complete copy of the change request definition, approved test procedures, and all test reports are needed. Copies of the recommended user manual changes and the baselined user manuals are also used. Recommendations of the completeness of the test results are provided to the system manager through the software manager to approve release for the next test activity, either independent testing, user testing or operational use.

As the needed documents are made available, each discipline in the software support activity and the user of the applicable system review them for validity and completeness in preparation for the formal test report review. The software manager then holds a formal review discussing each discipline's comments and prepares a recommendation to be presented to the system manager at the appropriate test readiness review or the acceptance of the change.

As each phase of the test process is completed, the software manager updates the utilization of resources and redistributes, as needed, the remaining cost and schedule parameters. A summary of this utilization and redistribution effort is presented to the system manager at each test readiness/acceptance review.

4.5.6.15 INDEPENDENT TEST READINESS REVIEW

This review will determine the acceptability of the software change package for release to an independent test agency for system performance testing (not acceptance).

Inputs to this review will consist of internal test plans and the results of the internal tests, i.e., the test reports and their evaluation. Also reviewed will be the change documentation package, to include users manual updates and training requirements. Successful completion of this review authorizes the establishment of the preliminary operational baseline and release of the baseline for performance testing.

Deficiencies identified during the internal tests should have been evaluated and corrected or if determined to be outside the scope of the current change requirements, communicate to the user and system manager for a determination of the handling of the deficiencies. The independent test agency should have

available for review by the software manager its test plan and should ensure the availability of resources to conduct the independent test. The system manager is responsible for approving the release of the change package for independent testing.

Significant cost and schedule impacts can occur due to the decisions made at this review. If it is determined that the change package is not ready for performance testing. The independent test agency may incur losses due to the realignment of resources necessary to accommodate the revised schedule. The implementation date requested by the user may be significantly affected.

4.5.6.16 USER TEST READINESS REVIEW

This review will determine the acceptability of the change package for user testing.

Inputs - Preliminary operational baseline, user manual updates, installation instructions, internal/independent test reports, user test plan/procedures, and the results from the Test Plan Reviews are used to determine readiness for user test. Output will be approval for the change package to enter testing by the user.

The system manager decides if the independent test phase is complete and the change implementation is ready for user test. The user is responsible for identifying the resources required to conduct the testing and must decide if the change package contains all of the resources needed for the conduct of his test.

Funding for user test should be provided by the user, with the level of test effort and compatibility with change development schedule coordinated with the software manager in previous activities. The software manager should provide a complete prototype change package for integration into the test system.

4.5.6.17 ACCEPTANCE REVIEW

This review will result in user acceptance of the change package. The user must be satisfied that the change fully meets the stated requirements. The user will certify his acceptance of the change for distribution to the field and forward that certification to the system manager.

Inputs are user test reports, deficiencies identified during test with plan for corrective action, complete change documentation package, user manual updates, implementation requirements and constraints and evaluation of the test reports by the software support activity. Outputs are user certification and system manager approval.

All deficiencies discovered during user testing will be reviewed to determine further course of action. All administrative and technical activities required for implementation of the change package must be complete. When this is accomplished and the user has advised the system manager of the certification of the change package, the system manager will accept the package and provide to the user for implementation. The user and software manager must come to an agreement on implementation schedule and distribution requirements. The authority for implementation of the fully developed change package lies with the user.

4.5.6.18 STATUS REVIEWS

In addition to the formal management reviews detailed in the foregoing there is an ongoing status review function which must occur to ensure against "surprises" at one of the major milestones or decision points. Of paramount importance an accounting of resources expended against programmed and time used against scheduled. The system developed to provide this oversight may be as formal as an automated management system or as informal as a weekly manager's review. Irrespective of form, however, the process used must be well understood by all involved, must be accurate enough to objectively measure critical parameters and must be tailored to the complexity of the system it tracks.

Inputs to the status review function consist of parametric data dealing with cost, schedule, percent completion, and the like. Inputs are defined based on the nature of the status accounting system, e.g., automated or manual. Outputs are in the form of variances from projected standards or norms and are used to flag management attention to unwanted situations.

Responsibility for routine status reviews rests with the software manager and it is incumbent upon him to analyze out of tolerance conditions, take corrective measures, and notify the system manager of problems beyond his control.

4.5.6.19 PRIORITY CHANGE PROCESS

In the event that priority change is identified certain formal management controls need to be bypassed to meet schedule constraints. As focal point for all system changes the system manager must classify any priority change with user input.

4.5.6.19.1 Task Identification and Control

The system manager must identify each priority software change to the software manager who in turn passes the change request on to the engineering organization to define and scope.

4.5.6.19.2 Change Requirement Review

After informal coordination with the user a change requirement is identified by engineering including initial design concepts/constraints and resource allocations. The software manager reports back to the system manager for go ahead. The system manager provides resources as needed.

4.5.6.19.3 Status Reviews

Informally, the software manager will check on progress of the design, implementation and test of the change.

4.5.6.19.4 Formal Testing

The magnitude, level, and complexity of testing for a priority change varies as a function of the time available to the maximum extent practical, internal and user test should be integrated. Based on the results of such integrated tests, the user, in consultation with the software manager, may waive independent user tests and proceed directly for formal release.

4.5.6.19.5 Management Clean-up

After release of the emergency change and acceptance by the user certain management documents need to be updated to reflect the change. The software manager is responsible to assure that these documents and the baselines they define are modified to reflect the new configuration.

4.5.7 TECHNICAL CONTROLS

4.5.7.1 SCOPE

This section contains the policy for the exercise of technical controls during the software change process. These controls fall into three general areas: change engineering, testing and software quality assurance.

At this time, this policy is limited to the change process for a single deployed system which is supported by a software support activity. Multiple system impacts and changes to the software made outside the SSA are not covered under this policy.

4.5.7.2 SOFTWARE ENGINEERING CHANGE PROCESS. The following paragraphs establish the policy for the software engineering change process functions. The functions are depicted in the several software change process functional model Figure 4.5-1. Each function establishes the policy required to preserve software design integrity of the product and to manage changes to software during operations and software enhancement of the software life cycle.

4.5.7.2.1 Change Notification Function. This function shall review the change request.

4.5.7.2.1.1 Inputs. The change request shall be reviewed to determine that all information is complete and in order that planning for the change can be accomplished.

4.5.7.2.1.2 Outputs. The output from this function shall provide the data for preparation of the preliminary technical package for the change and problem definition. Preliminary assignment of task and priority for the change shall be recommended to management. Notification shall be forwarded to SCM of implementing changes.

4.5.7.2.2 Preliminary Change Requirements Function. This function shall determine the change requirements based on the problem definition preliminary, task assignment, the priority of the change and the baseline package.

4.5.7.2.2.1 Inputs. The problem definition shall be reviewed as to impact on cost, schedule, resources, performance and baseline documentation. An analysis shall be with these data to determine if the change is a routine change, the change is an emergency change and if the change might require hardware change. Based on the analysis and problem definition, effective, retrofit and priority for the change shall be established and recommended to management.

4.5.7.2.2.2 Outputs. This function shall provide the design change concept, and an estimate of resource needs, the estimated cost to design and implement the change, the schedule for design and implementation of change, the expected performance and impact to status accounting.

4.5.7.2.3 Top Level Design. This function shall accomplish the top level design.

4.5.7.2.3.1 Inputs. The inputs to accomplish this function shall be based on the design concept, user's input, management changes and design deficiencies (trouble reports) received during various levels of testing. The design and document shall be in accordance with standards for software.

4.5.7.2.3.2 Outputs. The outputs of this function shall be the top level design for the change, updated documentation package, and details for preparation of test plans. In addition, identified problems/changes in status, schedule, and resources shall be reported.

4.5.7.2.4 Detail Design Function. This function shall accomplish the detail design.

4.5.7.2.4.1 Inputs. Based on the approved top level design and associated documentation, the detail design shall be accomplished. Furthermore, updates to detail design documentation shall be made upon receipt of deficiencies reports. The design and documentation shall be in accordance with applicable standards.

4.5.7.2.4.2 Outputs. The outputs of this function provide a performance package for detail design review, data for preparation of detail test procedures and code generation.

4.5.7.2.5 Code Generation. This function shall generate the code for the change.

4.5.7.2.5.1 Inputs. Based on the approved detail design, associated design documentation, and programming standards the code shall be generated for the change. Code deficiencies reported shall be evaluated and corrected.

4.5.7.2.5.2 Outputs. The outputs of this function provide inputs to the test function, and updates for documentation.

4.5.7.2.6 Test Deficiencies Evaluation. This function shall accomplish test deficiencies evaluation of module, module integration, system integration, independent performance, stress, and users acceptance testing.

4.5.7.2.6.1 Inputs. Inputs to accomplish function are provided by test deficiencies reports. Inputs shall be evaluated and analyzed to determine where the deficiencies are.

4.5.7.2.6.2 Outputs. From the evaluation and analysis of the test deficiencies, a report shall be forwarded to management control and the function or functions that shall accomplish the correction of the deficiencies.

4.5.7.3 TEST FUNCTION.

4.5.7.3.1 Scope of Test Requirement

4.5.7.3.1.1 Description. This function will address the technical aspects of testing the change within the management guidelines provided in order to produce test requirements.

4.5.7.3.1.2 **Inputs.** This function has the following inputs: task assignment from management; configuration baseline documentation packages; problem definition; and originator change notification/block change notification.

4.5.7.3.1.3 **Outputs.** The concept of test is the output from this step.

4.5.7.3.1.4 **This Step Precedes.** It is envisioned that this function spans the change concept determination functions. Since block changes will be accumulated, they will appear as only one change administratively.

This function must be complete prior to preparation of a test plan outline and prior to review of requirements baseline.

4.5.7.3.2 Test Plan Outline

4.5.7.3.3 Draft Test Plans

4.5.7.3.3.1 **Description.** This function will address the preparation of an outline delineating principal objectives, criteria, data requirements, analyses requirements, and resource requirements.

4.5.7.3.3.2 **Inputs.**

- a. Performance Package.
- b. Concept of Testing.

4.5.7.3.3.3 **Outputs.**

- a. Test Outline.
- b. Cost and Schedule Package.

4.5.7.3.3.4 **What This Function Precedes.**

- a. Management decision to allocate resources.
- b. Update of Requirements Baseline.

4.5.7.3.4 Draft Test Plans

4.5.7.3.4.1 **Description.** This function will be the preparation of test plans tailored to the top level design that addresses the subsequent levels of testing.

4.5.7.3.4.2 **Inputs.**

- a. Top Level Design Performance Package.
- b. Test Outline.

4.5.7.3.4.3 **Outputs.**

- a. Test Plans
 - 1. Module test plans
 - 2. Module integration test plans
 - 3. Systems integration test plans
- b. Evaluation Plans
 - 1. Code evaluation plans

2. User manual change evaluation plans

4.5.7.3.4.4 What This Function Precedes.

- a. Update of Top Level Design Baseline
- b. Commencement of Detail Design

4.5.7.3.5 Detail Test Procedures

4.5.7.3.5.1 Description. This function will address the detailed test execution procedures and evaluation procedure necessary to perform the myriad levels of software test and evaluation from guidance provided as input.

4.5.7.3.5.2 Inputs.

- a. Detail design performance package
- b. Test or evaluation plan
 - 1. Code evaluation plan
 - 2. Module test plan
 - 3. Module integration test plan
 - 4. System integration test plan
 - 5. User manual change evaluation plan
- c. Review approval

4.5.7.3.5.3 Outputs. Evaluation procedures or test execution procedure.

- a. Code evaluation procedures
- b. Module test execution procedures
- c. Module integration test execution procedures
- d. System integration test execution procedures
- e. User manual change evaluation procedures

4.5.7.3.5.4 What This Function Precedes.

- a. Update of system detail design baseline
- b. Update of user's manual

4.5.7.3.6 Test and Evaluation Execution

4.5.7.3.6.1 Description. This function addresses the execution of actual tests and/or evaluation required by the specified procedures.

4.5.7.3.6.2 Inputs

- a. Evaluation procedures or test execution procedure
 - 1. Code evaluation procedures
 - 2. Module test execution procedures
 - 3. Module integration test execution procedures
 - 4. System integration test executive procedures
 - 5. User's manual change evolution procedures
- b. New code or user's manual
- c. Review approval
- d. Management problem status guidance

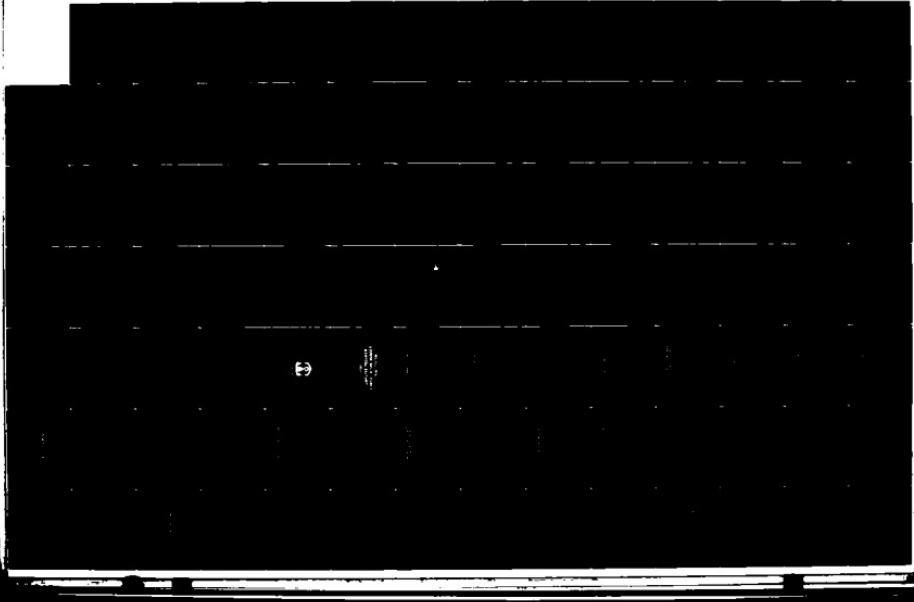
AD-A159 109

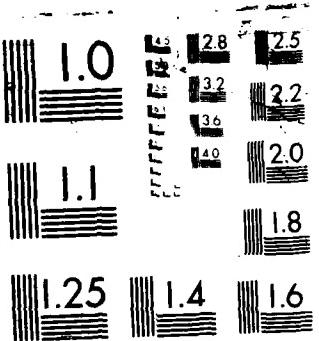
JOINT LOGISTICS COMMANDERS' WORKSHOP ON POST-DEPLOYMENT
SOFTWARE SUPPORT (U) JOINT POLICY COORDINATING GROUP 3/6
ON COMPUTER RESOURCE MANAGEMENT JUN 84

F/G 12/3

NL

UNCLASSIFIED





4.5.7.3.6.3 Outputs.

- a. Test and evaluation reports
 - 1. Code evaluation reports
 - 2. Module test report
 - 3. Module integration test report
 - 4. System integration test report
 - 5. User's manual changes
- b. Software change package
- c. Test and evaluation deficiencies
- d. Advise management of problems

4.5.7.3.6.4 What This Function Precedes. Release for independent system testing.

4.5.7.3.7 Independent System Performance Testing

4.5.7.3.7.1 Description. This function will address the testing performed on the entire system, both hardware and software, to determine the systems performance bounds and the impact of the change of established performance.

4.5.7.3.7.2 Inputs.

- a. Software change package
- b. Release approval for independent system testing
- c. Management guidance on project problems
- d. Changes to user manuals

4.5.7.3.7.3 Outputs.

- a. Systems software change package
- b. Test report
- c. System performance deficiencies
- d. Problems/changes in project elevated to management

4.5.7.3.7.4 What This Function Precedes.

- a. Approval for user testing
- b. Update of operational baseline

4.5.7.3.8 User Acceptance Test and Trial Installation

4.5.7.3.8.1 Description. This function will address the operational testing performed on the entire system by military personnel in an operational environment. The acceptability of the change to the user will be assessed.

4.5.7.3.8.2 Inputs.

- a. System software change package
- b. Release for user testing

4.5.7.3.8.3 Outputs.

- a. User test report
- b. User test deficiencies

4.5.7.3.8.4 What This Function Precedes

- a. Publication of documentation
- b. Release of change

4.5.7.4 SOFTWARE

4.5.7.4.1 Problem Definition

The problem definition function analyzes the change request in order to translate it into a system problem definition. The problem definition will form a basis for subsequent testing of the system to determine if the problem has been corrected. If the change request required modification due to an inability to define the problem, the originator will be contacted to redefine or clarify the change request.

4.5.7.4.1.1 Input to this function is the change request.

4.5.7.4.1.2 Outputs from this step include the status report. The problem definition, and a technical package consisting of the change request, notes, and the problem definition. A report shall be issued for status accounting. The report shall include the change request, the problem definition, and results of conversations (if any) with the originator.

4.5.7.4.1.3 This step must precede decisions on change and test requirements.

4.5.7.4.2 Design Change Concept Review

The design change concept, schedule, and resource requirements shall be reviewed to determine if the defined problem will be corrected within originator and management established constraints. The change will be prioritized using established guidelines.

4.5.7.4.2.1 Inputs shall consist of the preliminary technical package and the performance and schedule package.

4.5.7.4.2.2 Outputs shall consist of a cost and schedule package for management review, a requirements package for configuration management, and a technical package combining all these products. Status reports summarizing the review shall be submitted for status tracking.

4.5.7.4.2.3 This review produces critical management and configuration control inputs. It must be completed prior to any software design.

4.5.7.4.3 Technical Package Approval

The technical package shall be reviewed in accordance with the approved cost, schedule, and resource constraints established by management.

4.5.7.4.3.1 Inputs shall consist of the technical package and a management package consisting of cost, schedule, and resource approval and constraints.

4.5.7.4.3.2 Output shall be an approved technical package. Reports on the status

of the technical packages shall be generated for status tracking.

4.5.7.4.3.3 This step must be completed prior to the top level design review.

4.5.7.4.4 Top Level Design Review

The top level design shall be reviewed against the problem definition and other constraints contained in the technical package. Design problems shall be reported to management for resolution.

Once the top level design is approved, it will be merged with the technical package to form the top level design package.

Baseline changes from the approved top level design will be reported to configuration management.

4.5.7.4.4.1 Inputs shall consist of the performance package and the approved technical package.

4.5.7.4.4.2 Outputs shall consist of an approved top level design package and the top level design baseline changes.

4.5.7.4.4.3 This step must be completed prior to detail design commencement.

4.5.7.4.5 Test Plan Reviews

Test plans shall be reviewed against the performance established in the top level design package. The review shall answer two primary questions: Is the test satisfactory to show that the problem has been corrected; and is the test sufficient to detect undesired system behavior introduced by the change.

Problems encountered during this review shall be reported to management for resolution.

The approved test plan shall be incorporated into the top level design package to form the test plan package.

4.5.7.4.5.1 Inputs shall consist of the test plan and the top level design package.

4.5.7.4.5.2 Outputs shall consist of the approved test plan and the test plan package.

4.5.7.4.5.3 This step shall be completed prior to the development of detail test procedures.

4.5.7.4.6 Detail Design Review

The detail design shall be reviewed against the top level design (part of the test plan package).

Design deficiencies shall be reported to management.

Status reports will be issued to report design progress.

After the detail design is approved, changes to the detail design baseline shall be reported to configuration management.

4.5.7.4.6.1 Inputs shall consist of the performance package containing the detail design, and the test plan package.

4.5.7.4.6.2 Outputs shall be the approved detail design attached to the test plan package to form the detail design package and the detail design baseline changes.

4.5.7.4.6.3 This step shall be completed prior to code generation and the detail test procedures review.

4.5.7.4.7 Detail Test Procedures Review

The detail test procedures shall be reviewed against the test plan contained in the detail design package. The primary purpose of this review is to assure that the test procedures are necessary and sufficient to satisfy the test plan. A secondary purpose is to assure that the test procedures are realizable within established physical, resource and schedule constraints. If the detail test procedures are deficient a problem report shall be submitted to management for resolution.

4.5.7.4.7.1 Inputs shall include the detail test procedures and the detail design package.

4.5.7.4.7.2 Outputs include the approved test procedures and the detail test procedures package which combines the detail test procedures with the detail design package.

4.5.7.4.7.3 This step shall be completed prior to software testing.

4.5.7.4.8 Engineering Test Report Review

Engineering test reports shall be reviewed to determine when the approved testing has been satisfactorily completed and the software has been shown to be functioning properly. At this time, changes to user manuals shall also be reviewed for content.

The primary result of this step is to recommend the release of the software for independent system performance tests.

Reports documenting this release recommendation shall be submitted for status tracking.

4.5.7.4.8.1 Inputs to this step will consist of the detail test procedures package, the test reports and the recommended changes to user manuals.

4.5.7.4.8.2 Outputs from this step include a recommendation to management for release of independent system testing, the test reports and the operational baseline and manual changes.

4.5.7.4.8.3 This step shall precede independent system performance testing.

4.5.7.4.9 Release for User Testing

Test reports from independent system performance testing shall be reviewed to determine if the desired system performance has been achieved. If it has, a recommendation for technical release shall be issued for management action.

4.5.7.4.9.1 Inputs to this step are the previously approved test reports and independent system performance testing.

4.5.7.4.9.2 Outputs include the recommendation for release for user testing and the approved collection of test reports.

4.5.7.4.9.3 This step shall precede user installation and testing.

4.5.7.4.10 Final Test Report Review

User acceptance test reports shall be reviewed and combined with the overall technical package. The approved reports and baseline documentation shall be submitted to configuration management, and a recommendation for change release made to management.

4.5.7.4.10.1 Inputs will include the approved technical report package and the user acceptance test report.

4.5.7.4.10.2 Outputs shall include the baseline documentation and the final report package. A final report will also be issued for status tracking.

4.5.7.4.10.3 This step will precede change release.

4.5.7.5 EXTERNAL INTERFACE. Following deployment of systems, operational needs will dictate software interface which could not be anticipated from front-end design. Software change requirements may occur which require an enhanced performance which may dictate hardware change or expansion to accommodate implementation of user specified needs. When this determination has been made, software change may be developed but deferred for action/implementation until the hardware can accommodate such change.

4.5.7.5.1 The input for such a determination of hardware constraint may not occur until the scope of the change requirement is completed as a preliminary change design.

4.5.7.5.1.1 The output following a determination of hardware constraint or limitation shall be an exit from the model pending management solution and decision.

4.5.7.5.1.2 This action shall be on additional functional design box added to the model prior to establishing design change concept and estimate of resource needs.

4.5.7.5.2 Below the functional design box entitled "Review Design Concepts with originator and establish baseline and priority of Change" shall be added on interface box entitled "Other Systems Interop." Approval or disapproval of the proposed software change by such other systems shall be made by that system/manager. Approval allows continuation of change process by re-entering the exited box. Disapproval required return to functional box entitled "Determine Preliminary Change Requirements."

4.5.7.5.2.1 Interoperability requirements shall be established and agreed upon by

affected systems prior to the development box of "Perform Top Level Design" for this system.

4.5.7.5.2.2 Output shall be an agreed upon interface specification which affected system prior to the development box of "Perform Top Level Design" for this system.

4.5.7.5.3 There is a requirement to support training required to perform user acceptance testing of new/changed software. This shall be added to the model based upon final design when operator manuals can be developed and when training can be provided for designated testing personnel.

4.5.7.5.3.1 Immediately underneath the box entitled "General Code," a box shall be added to the model based upon final design when operator manuals can be developed and when training can be provided for designated testing personnel.

4.5.7.5.3.2 Input shall be a description of functional capability to be provided by the software change plus instructions for accessing and utilizing the information by operating personnel.

4.5.7.5.3.3 Immediately prior to the box entitled "Review Detail Test Procedure" a box shall be added entitled "Initiate Instructor and Key Personnel Training." Follow the new box with a box entitled "Train Operational Test Personnel."

4.5.7.5.3.3.1 Re-enter model on line labeled "Approved Test Procedures."

4.5.7.5.3.4 Exit model on line labeled "Approved Test Procedures."

4.5.7.5.3.4.1 Re-enter model on line labeled "Test Reports and Baseline Documentation for Technical Release."

4.5.7.5.3.4.2 Input to final manual page changes shall come from operating test personnel where they evaluate printed operator instruction at the same time they evaluate system performance.

4.5.8 PRE-DEPLOYMENT RESPONSIBILITIES OF THE SOFTWARE SUPPORT ACTIVITY (SSA)

4.5.8.1 GENERAL

The SSA has a vested interest in ensuring that proper attention is directed towards the software design and development process and the support system environment early in the acquisition cycle. The program manager must select the SSA early in the planning cycle. The SSA should be actively involved in all phases of the acquisition cycle to optimize its capability to perform life cycle support. The utilization of the SSA in RFP production, design reviews, audits, IV&V , and configuration control will enhance project effectiveness, reduce the effort required to prepare life cycle support and ensure a smooth transition of the product software from the developer to the government. The SSA is responsible for preparing the support organization to ensure funding, training and procedure plans are in place at transition. Specific responsibilities to be assigned to the SSA for each acquisition cycle phase are addressed in the following paragraphs.

4.5.8.2 CONCEPT PHASE

4.5.8.2.1 Planning Activities

The SSA must begin preparation for the support role as soon as the SSA is designated. SSA key personnel must prepare a Computer Resource Life Cycle Management Plan (CRLCMP) which will become a living document, revised as needed throughout the life cycle. Staffing, based on complexity of the system to be supported must be justified. Training for the staff must be identified based on the realities of the available manpower pool skill level. Sufficient staff must be assigned to the project immediately to ensure continuity throughout the acquisition phase and into the support phase.

The Software Support Environment (SSE), including physical facilities, computer and other GFE equipment, communications, software tools, and security requirements must be identified.

Strong justification for each of the requirements must be developed by the SSA staff and submitted as refinements to the budgetary process.

4.5.8.2.2 Solicitation Preparation Activities

The SSA shall assign personnel to the procurement team to ensure that the solicitation (RFP) includes all requirements necessary to acquire supportable software. Issues such as equipment and software compatibility, future expansion or change needs, deliverable support software, hardware, documentation, interchangeability requirements and legal issues such as proprietary software must be considered. The SSA personnel must ensure that supportability requirements will be given significant weight in the selection process. The SSA shall bring the full weight of lessons learned (including system commonality vs system unique features) to bear on this activity. Interface requirements with other support elements such as ATE, training simulator, or ground support for embedded computers shall be included in this activity.

4.5.8.2.3 Software Support Activity (SSA)

The SSA will provide membership to the source selection evaluation process and will be a voting party in its recommendation to the source selection authority. The SSA representative(s) shall evaluate each part of the proposal which directly or indirectly affects the future supportability of the system being procured.

4.5.8.3 DESIGN AND DEVELOPMENT PHASE

During the design and development phase of embedded computer systems software, the SSA shall be assigned responsibility for ensuring that all software end items are produced in accordance with governing specifications. Designation of the SSA as the development activity's verification and validation agent and as the agent responsible for other monitoring and auditing roles during this phase satisfies this requirement. This designation has the desired effect of providing a highly motivated but independent source to act as the project's IV&V agent. In addition, the SSA must also be responsible for completing preparations for assuming post deployment software support as specified in the approved CRLCMP. Specifically the SSA responsibilities entail the following activities.

4.5.8.3.1 VV Activities

The SSA shall undertake verification and validation activities independent of the development activity design and development process, of sufficient magnitude to provide a thorough and unbiased evaluation of the product(s) under development. The degree of independence and scope of the V&V effort will vary depending on product complexity; however, the SSA or IV&V contractor under the direction of the SSA shall as a minimum perform the following IV&V functions.

1. Review software development plans, CM and QA plans.
2. Review all design documents.
3. Perform code analysis.
4. Review test plans and specs.
5. Review test results.
6. Report all results to the PM along with recommended corrective actions.

4.5.8.3.2 Support Environment Establishment Activity

The SSA will assume responsibility for preparation of the support environment required by the CRLCMP to provide all post deployment support for the embedded computer software and firmware. These activities include acquisition and installation of hardware and software, training of SSA personnel, developing procedures and technical controls and the procuring of appropriate contractor support personnel. These activities shall be accomplished during the Design and Development phase to ensure availability of a SSA support facility coincident with the deployment (responsibility transfer) of the system.

4.5.8.3.3 Configuration Management Monitoring Activity

Configuration Management of the software becomes one of the major responsibilities of the SSA upon product transition; therefore, the SSA must become cognizant of an expert in the configuration management requirements of the system software. The SSA must insure that, upon transition, configuration management can be assured. The SSA shall establish a software configuration control group to monitor all software Configuration Management activities during the design and development phase to ensure effectiveness and also to become thoroughly familiar with the CM data base prior to assuming post deployment life cycle responsibilities. During this period the SSA will develop suitable configuration management procedures to be activated upon deployment of the system. The SSA shall serve as a member of the Configuration Control Board and will monitor the disposition of all approved changes.

4.5.8.4 ACCEPTANCE PHASE ACTIVITIES

4.5.8.4.1 Supportability Audit Activities

The SSA shall be responsible for conducting a Supportability Audit for the purpose of certifying all system software, as delivered by the development activity, can be fully supported in the SSA environment. Successful completion of the Supportability Audit constitutes a further refinement of the Operational Baseline (OBL) in that programs generated at the SSA site are certified as physically and operationally equivalent to the OBL. The SSA conducted Supportability Audit shall consist of the following activities.

- a. A physical audit of all system and support software delivered by the development activity.
- b. The exercise of all support software tools against the operational baseline library in accordance with PM approved test plans.
- c. The building of all program variants using SSA facilities and personnel.
- d. The exercise of all program variants in target computers in accordance with SSA written, PM approved test plans.
- e. The reporting to the PM of the SSA's assessment of the software systems supportability.

4.5.8.4.2 Acceptance Test Activities

The SSA shall serve as an official member of the acceptance test team. Participation in these activities has the dual purpose of supporting the PM in the acceptance process and of providing the SSA with a true picture of the software being delivered.

PANEL E - CHANGE PROCESS

CO-CHAIRMAN: Mr. J. (Joe) Black (912) 926-5948
WR-AFLC/MMRR A/V 468-5948
Robins AFB, GA 31098

CO-CHAIRMAN: Mr. J. (Jack) Cooper (703) 276-2826
CACI Inc., Federal Penthouse
1700 N. Moore St.
Arlington, VA 22209

MEMBERS:

A	Ms. D. (Donna) Brock USA MICOM (DRSMI-RGG) Redstone Arsenal, AL 35898	(205) 876-8309 A/V 746-8309/ 4440
A	Mr. B. (Bob) Niveson USA TECOM/DRSTE-AD~S Aberdeen Proving Ground, MD 21005	(301) 278-2775 A/V 283-2775
A	Mr. D. (Dean) Gunn USA TRADOC (ATCD-CB) Ft. Monroe, VA 23651	(804) 727-3271 A/V 680-3271
A	Mr. T. (Troy) Madison Field Artillery School and Center (ATSF-CT) Ft. Sill, OK 73503	(405) 351-6089 A/V 639-6089/ 4867
N	Mr. G. M. (Jerry) Wrout Pacific Missile Test Center (1220)	(805) 982-8865 A/V 351-8865
N	Mr. W. D. (Don) Balmer Fleet Combat Direction Systems Support Activity San Diego, CA 92147	(619) 225-7021 A/V 933-7021
N	LCDR M. (Mike) Gehl (JLC-CRM-CSM Chairman) Naval Electronic Systems Command Washington, DC 20363	(202) 692-8484 A/V 222-8484
MC	MAJ J. H. (Joe) Burney Marine Corps Tactical Systems Support Activity Camp Pendleton, CA 92055	(619) 725-2421/ 2288 A/V 993-2421/ 2288
AFLC	Mr. G. (George) Montagno TAWC/EWER Eglin AFB, FL 32542	(904) 882-8814 A/V 872-8814
AFLC	Mr. G. (Gaylen) Pederson OO ALC/MME Hill AFB, UT 84053	(801) 777-7108 A/V 458-7108

PANEL E (cont'd)

AFLC	Mr. Mark vandenBroek ICG/CFE AFB, OH 45433	(513) 257-6751 A/V 787-6751
AFSC	Ms. K. (Karen) Bausman ASD/YWE Wright-Patterson AFB, OH 45433	(513) 255-2885 A/V 785-2885
AFSC	LTC R. H. (Robert) Brown SAC/LGY Offutt AFB, NE 68113	(402) 294-2892 A/V 271-2892
N	Mr. C. (Cal) Carrera Veda Inc. 1317 Del Norte Rd. Camarillo, CA 93010	(805) 485-6535
N	Mr. R. D. (Bob) Buck IBM Federal Systems Division 9500 Godwin Drive Mail Code 887-035 Manassas, VA 22110	(703) 367-3266
A	Dr. W. (Wilson) Talley University of California-Davis Lawrence Livermore Laboratory P.O. Box 808 L794 Livermore, CA 94550	(415) 422-9787
A	Mr. L. (Lee) Whitley TELOS Federal Systems P.O. Box 33099 Ft. Sill, OK 73503	(405) 355-9280
AFLC	Mr. L. D. Parriott TRW 1 Space Park Redondo Beach, CA 90278	(213) 217-3077
AFSC	Mr. Harold David Hall Texas Instruments, Inc. P.O. Box 405 MS3407 Lewisville, TX 75067	(214) 462-5497
O	Ms. W. (Wilma) Osborne NBS Institute for Computer Science & Technology Bldg. 225, Rm. B266 Washington, DC 20234	(301) 921-3545
	Ms. J. (Joan) Spaulding FAA 800 Park Crest Dr. Silver Spring, MD 20910	(202) 426-8634

PANEL E (cont'd)

NTEC	Mr. P. (Paul) Byrley Naval Training Equipment Ctr (N-213) Orlando, FL 32813	(305) 646-5354 A/V 791-5354
	Dr. Kirt Fisher Computer Science Corp. 6565 Arlington Blvd Falls Church, VA 22046	(703) 237-2000 ext 6966

ORLANDO I

FINAL REPORT

PANEL F

SOFTWARE CONFIGURATION MANAGEMENT

MARCH 15, 1984

CO-CHAIRPERSON:

Antonia D. Schuman (Toni)
TRW Systems Group
1 Space Park, Bldg. 134
Room 6079
Redondo Beach, CA 90278
(213) 217-4079

CO-CHAIRPERSON:

Mr. C. (Cal) Showalter
Navel Air Systems Command
(AIR-543C)
Room 620, JP-2
Washington, DC 20361
(202) 746-0650

TABLE OF CONTENTS

Paragraph		Page
4.6	SOFTWARE CONFIGURATION MANAGEMENT	4-6-1
4.6.1	Introduction/Objective	4-6-1
4.6.2	Scope	4-6-1
4.6.3	Approach	4-6-2
4.6.3.1	Preparation	4-6-2
4.6.3.2	Panel Organization and Operation	4-6-3
4.6.3.3	Issues	4-6-3
4.6.4	Discussion	4-6-3
4.6.4.1	Subpanel A: Baseline Definition	4-6-3
4.6.4.1.1	Turnover Products	4-6-4
4.6.4.1.2	Transition of CM From Developer to SSA	4-6-5
4.6.4.2	Subpanel B: Configuration Management Scope and Requirements	4-6-7
4.6.4.2.1	Definition of PDSS Configuration Management Scope	4-6-7
4.6.4.2.2	PDSS Configuration Management Interface and Influence . .	4-6-9
4.6.4.2.3	Computer Software Configuration Item (CSCI)	4-6-11
4.6.4.2.4	Computer Program Identification Numbers (CPIN) - USAF . .	4-6-12
4.6.4.3	Subpanel C: Security	4-6-13
4.6.4.4	Subpanel D: Configuration Status Accounting	4-6-16
4.6.4.4.1	What CSA Information is Stored, Tracked, and Reported? .	4-6-17
4.6.4.4.2	How Should CSA Information be Stored, Tracked, and Reported	4-6-19
4.6.4.4.3	To Whom Should CSA Information be Available?	4-6-20
4.6.4.4.4	Where Should CSA Information be Stored?	4-6-20
4.6.4.4.5	What Developer Contractual Constraints are Required . . .	4-6-21
4.6.4.4.6	Summary	4-6-21
4.6.5	Recommendations	4-6-22

ATTACHMENTS

4-6-I:	Page
4-6-I-1 Guide for Selecting Configuration Items (CI's)	4-6-25
4-6-I-2 Computer Program CI (CPCI) Consideration	4-6-26
4-6-I-3 Computer Program CI (CPCT) Consideration (cont.)	4-6-27

APPENDICES

4.6 SOFTWARE CONFIGURATION MANAGEMENT

4.6.1 Introduction/Objective

Defining the role of Configuration Management in the post deployment phase has been tried a number of times with limited success. Each of the three services approaches PDSS Configuration Management (CM) from slightly different perspectives. Possibly this has resulted from an attempt to apply reasonably mature hardware CM practices to software. The elements of CM: configuration identification, control, and status accounting are all as important in the software world as in the hardware world. Configuration control is even more important when the software is firmware in a Read Only Memory device. Changes to firmware installed in a piece of equipment could change the hardware configuration of the equipment.

One normally thinks of CM starting with a contract award and continuing through to the end of the system development cycle. One rarely thinks of CM beginning at the start of the development cycle and ending...maybe never ending...as long as the system is deployed, maintained, modified and enhanced.

However, if it is indeed CM's responsibility to maintain the system configuration from beginning to end, then end could be infinite.

CM is a primary focal point for communication within the acquisition program, the support functions, the customer and the user. It identifies the system configuration at specified points in the software life cycle, processes the changes to the established configuration - it controls and communicates those changes, it accounts for the status of the configuration and the changes that have been made and finally audits the delivered products to ensure that what was designed is what was built.

Maintaining configuration control in the deployment phase of a system's life cycle is vital. Changes are a continual process, and mission effectiveness must not be compromised by incorrect identification of the embedded software or inadequate controls over the change process from request to release.

Then, what is CM's role in the post deployment environment? It is simply this. It carries on from where it left off: it is a continuance of the CM function in the development of the system utilizing the deliverable products (per SOW and CDRL) as a basis for handling corrective changes, modifications and enhancements to the system's hardware and software.

4.6.2 Scope

The issues which the panel addressed all related to the scope of software configuration management in the post-deployment phase. The role and influence of the PDSS agency in the development phase (to make CM in the deployed phase easier) was also addressed in the context of DoD 5000.29 and MIL-STD-SDS.

Desired products were to be:

- * Baseline definition and recommendation for dealing with multiple, parallel baselines
- * CM transition plan
- * Recommended configuration identification and numbering system with policy statement on the need (pro or con) for commonality
- * Recommended policy statement on the role of the software support agency in the development phase
- * Policy recommendation on storage and tracking methods
- * Policy recommendation concerning software security processes, classification determination, verification, handling procedures, etc., for inclusion in DoD/individual services security manuals, directives, etc.

During the Workshop, the Configuration Management Panel addressed all questions on the Charter with particular emphasis on the expected products. However, we digressed slightly from the original charter and stated objectives (see Appendix D) because of the magnitude of some of the issues and the limited time available. Specifically, the Security Subpanel determined that the issue was larger than could be resolved in three days and chose to highlight the critical questions which needed JLC attention.

The Subpanel on Storage and Tracking Methods addressed the more general questions of configuration status accounting which include, besides tracking and storing CM data, reporting, configuration control and auditing.

4.6.3 Approach

4.6.3.1 Preparation

Prior to the Workshop, all participants received copies of the Charter and a questionnaire on which they were asked to provide comments on the Charter and select a subpanel of interest. They were also invited to address the full panel on relevant topics. Four briefings were given:

- a. Baseline Software Configuration Management: An Automated Approach. Naval Training Equipment Center Software Support Facility. Presented by John Ruckstuhl.
- b. USAF Computer Program Identification Number (CPIN) System (AFLC). Presented by James Wagoner.
- c. Air Force Data Systems Design Center Mission Briefing. Presented by Al Mayhan.
- d. Operational and Maintenance Configuration Management (Procedures). Presented by Ron Berlack, Sanders Associates.

Copies of the briefing charts are in Appendix C. Material of general interest was also collected as noted in the bibliography (Appendix B).

4.6.3.2 Panel Organization and Operation

The list of panel members is given in Appendix A. The panel divided into four subpanels based on the four principal issues in the Charter:

- * Baseline Definition
- * Scope and Terminology
- * Security
- * Storage and Tracking Methods

The full panel met every morning and evening to exchange preliminary findings and cross-pollinate. Several deep-seated concerns about the validity of MIL-STD-SDS and associated revisions to other MIL-STDs were raised. Some of the critical differences between the services also contributed to the liveliness of the general discussions. In the end, each subpanel addressed every question in the Charter and produced reports which met the basic objectives of the Workshop. Section 4.6.4 contains the subpanel reports.

4.6.3.3 Issues

The issues to be addressed by the panel are given in the Charter (Appendix D).

4.6.4 Discussion

The following paragraphs contain the individual reports of each subpanel. Specific issues are covered in detail, with analytical discussion, conclusions and recommendations for each topic.

4.6.4.1 Subpanel A: Baseline Definition

The subpanel consisted of:

Ms. Frances Soskins (Chair)
Mr. Max Bushey (Recorder)
Mr. Gerry Anderson
Mr. Joe Faranellio
LTC Dusty Rhoades (first day)
Mr. Al Mayhan
Mr. David Maibor

We addressed the questions provided in advance of the workshop. A fundamental conclusion reached relative to the "operational baseline" was that it is not appropriate for the configuration management organization to determine when a baseline system becomes operational. Our products are based on the assumption that the PDSS activity begins at a point defined for reasons unrelated to configuration management.

Our products consist of three reports:

- a. What products should be turned over from the developing activity to the support activity?
- b. How should the transition of configuration management from the development activity to the SSA be controlled?
- c. How to deal with multiple parallel baselines.

4.6.4.4.1 Turnover Products

4.6.4.1.1.1 Background and Discussion

The software support activity (SSA) performing post deployment software support (PDSS) manages the product baseline from the point of transfer when the SSA formally accepts responsibility and accountability from the development activity. The SSA updates the product baseline as modifications are made to the system software throughout the remaining life of the system.

4.6.4.1.1.2 Conclusion

In addition to the software itself in all forms and media in which it exists, the following should be turned over from development activity to SSA:

Engineering Documentation

System/Segment Specification
Software Requirement Specification
Interface Requirement Specification
Software Product Specification
Unit Development Folder*

Test Documentation

Software Test Plan
Software Test Document
Software Test Procedures
Software Test Report

Management Documentation

Computer Resources Life Cycle Management Plan
Software Standards and Procedures Manual
Software Development Plan*
Software Configuration Management Plan*
Software Quality Assurance Plan*
Version Description Document

Support Documentation

Operational Concept Document
Computer System Diagnostic Manual
Software Users Manual
Computer System Operator Manual
Firmware Support Manual
Software Programmers Manual

*Optional

Note that in some contracts the Software Development Plan contains the Software Standards and Procedures, whereas in others it is a separate document. The Standards and Procedures are not optional and, if part of the Development Plan, then the Development Plan becomes not optional.

4.6.4.1.1.3 Recommendation

Items identified in 4.6.4.1.1.2 should be reviewed for consistency with the planned configuration management approach. They should all be turned over in magnetically readable forms, if available, as well as hard copy.

4.6.4.1.2 Transition of CM From Developer to SSA

4.6.4.1.2.1 Background and Discussion

Transition of configuration management responsibility from the Development Activity to the SSA shall occur during the Production and Deployment Phase. It usually occurs during deployment, potentially in the interval during which production and deployment activities overlap. Before transition, the Development Activity shall remain responsible for management and control of computer resources, including computer software. During this time the SSA may be tasked to support the Development Activity.

4.6.4.1.2.2 Conclusion

A plan should be developed to transition configuration management responsibility from the Development Activity to the SSA.

4.6.4.1.2.3 Recommendation

The Transition Plan should be developed by both the Development Activity and the SSA and should include:

By the Development Activity

- a. Identification of scheduled dates for computer software delivery to the SSA.
- b. Identification of the product baseline to be transferred. This is to include all components.
- c. Disposition status of open problems and assigned responsibility for their resolution.

- d. Identification of the required configuration management support equipment, associated training, and applicable documentation.
- e. Status of completion of the system, subsystem or functions that impact computer software.
- f. Status of Documentation to be transferred.

By the SSA Activity

- a. Identification of plans for the conditions and contingencies which may dictate possible revisions to the scheduled delivery date.
- b. A description of how the PDSS Configuration Management Plan from the CRLCMP will be adopted.
- c. Provision for Configuration Management training.
- d. A plan for the audit of the software transition package (product baseline).

The Development Activity will be responsible for the production of the Configuration Management Transition Plan.

4.6.4.1.3 Multiple, Parallel Baselines (MPB)

4.6.4.1.3.1 Background and Discussions

MPB situations occur:

- a. When a product baseline is deployed to an operational environment while further development and/or modification to the product baseline is in process after the PDSS.
- b. When more than one version is deployed to an operational environment.
- c. When both of the above exist concurrently.

4.6.4.1.3.2 Conclusions

The essence of multiple baseline management is the application of standard configuration management principles (identification, control and status accounting) to each baseline.

4.6.4.1.3.3 Recommendation

Include in the Computer Resource Life Cycle Management Plan provisions for handling multiple, parallel baselines by establishing an estimate of the probable proliferation of baselines, establishing configuration management resource requirements in accordance with that estimate, and monitoring actual baselines and associated activities.

Estimates need to be revised as necessary throughout the life cycle, and changes should be implemented to handle increased complexity as soon as it appears that they will be needed (not delayed).

4.6.4.2 Subpanel B: Configuration Management Scope and Requirements

The subpanel consisted of:

Mr. Ron Berlack (Chair)
Mr. Norm Taupeka (Recorder)
Mr. Leo Gonzalez
Mr. Bob LaVergne
Ms. Dreama Fumia
Mr. Lew Eichert
Mr. James Wagoner
LTC Dusty Rhoades (second day)

Four major issues were addressed:

- a. Definition of PDSS CM Scope over individual systems and in regard to related and interfacing systems
- b. PDSS CM interfaces and influence throughout the life cycle
- c. Definition of computer software configuration items, who identifies them and when
- d. Computer program identification numbers: should there be a DoD standard?

Conclusions and recommendations on these issues were reached following considerable discussion.

4.6.4.2.1 Definition of PDSS Configuration Management Scope

The role of PDSS in system life-cycle management has not been well recognized or articulated as evidenced by incomplete planning for PDSS for systems as they enter the production/deployment phase. In addition, the practice and criticality of configuration management to a large degree is not always well understood or appreciated by the development, support and operational communities. As systems are being fielded and made operational with greater amounts of software, the continuing need for software change necessitates the expenditure of vast amounts of resources for increasingly complex systems. Even more critical is the ever increasing need to make accurate and timely changes for these fielded systems. Thus, in order to accomplish the fielding of reliable software for operational systems, the practices of configuration management during the deployment phase must be maintained and strengthened in a cost-effective manner.

4.6.4.2.1.1 Discussion

The configuration management of an automated military system is complicated by multiple interfaces, computers, displays, weapon mix and software components.

Managing the configuration of software when various versions are deployed compounds the problem and increases the complexity of the identification of each software version.

The interactive deployment of automated systems creates hardware diversity. Software changes compound this diversity. The relationship of the development program management offices to the PDSS agency under such conditions is not well delineated.

PDSS, in reality, is the continuation of the software development process. User modifications and enhancements to counter new threats, changes in doctrine, and introduction of advanced technology, fosters the evolutionary growth and increases capability in systems having embedded computer resources. In addition, errors requiring correction, interface and protocol conversions, modifications and enhancements are a part of PDSS. All of the above aspects therefore require configuration management.

During the development process the number of deliverables produced in the SOW per the CDRL serve as a basis for implementing CM in the PDSS phase. In addition, other products of the development phase such as software tools are required to strengthen the ability of the PDSS activity to effectively manage the software configuration.

Configuration management of interfaces among systems, especially when they are in different phases of the life cycle or are maintained by different agencies, services or nations, is a perplexing and ill-defined area with overlapping and sometimes conflicting responsibilities whose scope needs to be defined for the PDSS phase.

Within a given service, two or more systems having requirements for interface and interoperability must retain compatibility. CM is essential if these systems are to remain as such in a dynamic environment.

In addition, between and among the services, there is a need to interoperate compatible systems. With respect to NATO allies, a higher level of interoperability is required. To ensure interoperations of these systems, suitable levels of CM are required.

Another important aspect of CM is the need to assure appropriate interface and cooperative efforts with the developer(s) and other PDSS activity(ies) in order to influence the related generation of viable, deliverable products to the operational environments. Specifically, the software support environments, including compilers, link loaders and other tools, should also be configuration managed in order to simplify the transfer of the development software products between PDSS activities.

4.6.4.2.1.2 Conclusions

The Scope of CM in PDSS for Given Systems. PDSS configuration management should be a continuum of the CM function in the development of the system, utilizing the deliverable products (per SOW/CDRL) as a basis for handling corrective action changes, modifications and enhancements to the software.

The Scope of CM in PDSS for a Network of Related Systems. The scope should include the review and identification of impact on the subsystem and system interface, as well as the integration and interoperability of the interfacing systems.

4.6.4.2.1.3 Recommendations

- a. The JLC/CRM should initiate action to incorporate provisions for PDSS CM in applicable regulations, directives, MIL standards, and guidebooks to ensure the recommended continuation of configuration management in the PDSS phase.
- b. It is strongly recommended that the proposed joint regulations and procedures for configuration management of systems (HW & SW) be continued by the software support activity as transferred from the development phase to the deployment phase.
- c. Further, it is recommended that configuration management encompass all related, interfacing systems at all appropriate levels and be rigorously enforced.

4.6.4.2.2 PDSS Configuration Management Interface & Influence

Presently, PDSS/CM interface to the developer's CM activity as well as inter/intra service PDSS activities is not defined nor practiced. In addition, PDSS/CM influence during the development cycle is nonexistent.

To date, little attention has been paid to defining and implementing a PDSS interface into the development activity such as is done for Integrated Logistics Support (ILS). Little is to be found in current MIL-STDS, directives or regulations pertaining to PDSS planning, interface or a path for any level of PDSS activity influence to enable a viable operation during the deployment phase. Thus, PDSS has little or no input or right of concurrence in what configuration a given product will be developed or how the transfer will take place.

4.6.4.2.2.1 Discussion

The configuration management of embedded computer systems software during the PDSS phase must be a planned element throughout a system's life cycle.

PDSS configuration management must be considered during the development phase. As a continuum of the configuration management in the development phase, PDSS configuration management must take an active role in the initial development phase as well as possible subsequent interfacing development to assure optimal standardization of resources, techniques, and products and to provide for an effective transition of configuration management functions.

A single, integrated configuration management methodology for a suite of embedded computer system software is essential in the PDSS phase. The software in many complex weapon systems controls and integrates the functions of multiple subsystems to achieve the system's performance

requirements, and, in many cases, the software will even define the system configuration. Thus, the software configuration management must be a coherent subset of the parent system's configuration management to assure the maintenance of the essential hardware/software integration.

The configuration management of a software suite must also provide vital links with the configuration management of interfacing networks or systems to assure continued interoperability.

PDSS configuration management must be enhanced with increased attention and initiatives to achieve these goals.

4.6.4.2.2.2 Conclusions

If PDSS/CM is to be effective, then it must interface with and be part of, as a minimum, the following activities during the development phase:

- * Established Interface Control Working Group (ICWG)
- * The Software Configuration Control Board (SCCB)
- * Formal Design Reviews: SDR, PDR, CDR
- * Developer Internal Design Reviews
- * Review of Upper Level Testing
- * FCA, PCA, FQR

In addition, the PDSS CM activity must be able to ensure that the development CM activity has the deployment phase delivery as its primary objective in order to ensure an orderly and effective transfer of CM from one phase to the other. It also must have the ability to assure that interoperability factors are defined and in place, that applicable standardization of resources have been provided, such as CM automation tools, in order to provide development - deployment integrated CM activity.

4.6.4.2.2.3 Recommendations

The following paragraph has been submitted for inclusion in the MIL-STD-SDS draft of 31 August 1983:

The contractor shall specify PDSS parameters appropriate for the support of all the delivered and deployed software as part of the S/W Development Plan. These parameters, as a minimum, shall include the requirements for a support facility, work force mix, IV&V, cost of ownership, support environment and configuration management with details for change processing. These assessments, by necessity, are considered iterative in nature, shall be reviewed at CSCI design reviews and provide the basis for a PDSS plan to be implemented not later than establishment of the product baseline.

It is recommended that the JLC/CSM support this inclusion.

It is further recommended that appropriate interface guidance be included in applicable directives and regulations, and that a PDSS CM plan data item description be generated for implementing CM during the PDSS phase. Such a plan must provide for pre and post CM action as well as the interfacing that will be carried out with other PDSS activities in order to achieve appropriate CM for multi system deployment.

If these recommendations can be achieved, then the PDSS activity can and will be most effective and hopefully a persuasive influence over the complete system life cycle. This must also be concurrent with appropriate funding for optimum performance.

4.6.4.2.3 Computer Software Configuration Item (CSCI)

The basic question addressed: What is a CSCI and who defines it and when?

4.6.4.2.3.1 Discussion

CSCI designation is a fundamental building block for PDSS. CIs are a unit of management during development and support and are usually the level of effective management visibility throughout the life cycle. CI designation usually results in the following: separate development and product specifications, discrete identification, separate design reviews, testing, status accounting records, etc. Since CI definition and selection drives a significant number of attendant activities and products over the life cycle, care and good judgement must be exercised in the selection process.

During the systems engineering process begun in the concept exploration phase, elements of the systems are identified, requirements (functions) are allocated to them, and trade-off analysis performed to optimize the system as a whole. These system elements become elements in the work breakdown structure (WBS), but more importantly, when Type B design-to specifications are written for these elements on the basis of functional requirements allocated to them, they become configuration items. This results in a natural correlation between the WBS and the specification tree. As a result of the systems engineering process, the system is partitioned and its elements identified as manageable units, CIs.

As development proceeds and the design is iterated, understanding of the system and its functions evolve, resulting in changes in both the number and size of CIs, as well as in the functions allocated to them. Configuration management is an explicit recognition of the need to identify (document), control and maintain status of the process.

4.6.4.2.3.2 Conclusion

CI selection, both CSCIs and hardware CIs, is a product of the systems engineering process begun in the concept exploration phase and usually completed in conjunction with the system design review in the demonstration/validation phase. CI selection is an iterative process composed of inputs reflecting all program requirements and decided on the basis of both technical and management concerns.

4.6.4.2.3.3 Recommendations

MIL-STD-483, Notice 2, Appendix 17 (XVII), is a guide for selecting CIs. The proposed MIL-STD-483 changes included in the SDS "package" significantly weakens the guidance for selecting CPCIs (CSCIs). Therefore, the guidance in Notice 2 must be retained and changes to Appendix 17 notices confined to the administrative, i.e. - CSCI vice CPCI. Additionally, all JLC policy guidance on CI (CSCI) definition and selection should be consistent with that presently in Notice 2, MIL-STD-483. (See Attachment 4-6-I)

4.6.4.2.4 Computer Program Identification Numbers (CPIN) - USAF

Is it desirable for all DoD to settle on a standard or common computer program identification number (CPIN)? Can or should the CPIN fit into the hardware automated identification systems, and what is the content (not format) of a CPIN record?

The configuration management process requires that each configuration item, specifically each computer program configuration item (CPCI), be identified, that changes to the CPCI be controlled and managed, and that status accounting be maintained for the baselined CPCI and each change during the life cycle of the CPCI. Today there is no standardized number used in DoD for the identification of a CPCI. The CPCIs have been identified with a variety of numbers, such as manufacturer's part numbers, national stock numbers, technical order numbers, project numbers, and specially developed numbers. These CPCIs and related descriptive information are usually not indexed or announced in a publication. In most situations, users requirements for a CPCI and the number of copies required are maintained by the individual software manager. The distribution of a baselined CPCI is usually a "pull" type distribution where the user must requisition the CPCI through supply or equipment channels. Follow-on distribution of updates to the CPCI is incomplete. Few management reports are available and are usually compiled from manual records and documents on an "as required" basis.

4.6.4.2.4.1 Discussion

The Air Force has an automated CPIN data system centrally controlled and operated at Oklahoma City Air Logistic Center, Tinker AFB, OK. This system supports all embedded computer systems (ECS), software managers and software users within the US Air Force, and foreign governments supported through the Security Assistance Program. The CPIN system assigns a unique, sight recognizable identification number to each CPCI and related engineering documentation package. The CPIN identifier is incremented with a revision number each time the CPCI is updated. CPIN version numbers are assigned to CPCIs used on multi-weapon systems or support equipment. Consolidated indexes (CPIN compendiums) and cross-references are produced in microfiche, and copies are distributed monthly to software managers and users.

The CPIN system collects, stores and maintains data elements related to the CPCI, the software managers, CPCI changes, Configuration Control Board's approval dates, cost information, applicable weapon system and subsystem, control computer, support computers, supporting programs, programming language, type of deliverable media unit, contractor information, user computer operator manual, and a complete list of CPCI related engineering documentation. The system collects, stores and maintains software user requirements for each CPCI and produces mailing labels and software media labels for the "push" distribution of CPCIs (and changes) to each established software user. The system also produces management information products and reports for various levels of software managers and users.

4.6.4.2.4.2 Conclusion

A "CPIN" type system is necessary to achieve a standard approach for software identification, tracking CPCI changes, status accounting information, and to ensure "push" distribution of CPCIs and subsequent changes

to ECS software users for deployed or operational weapon systems and support equipment. Tri-service application is achievable through the use of common data elements. Also, the "CPIN" numbers should be assigned and controlled through a data systems for software, and should not be included in existing hardware or supply systems.

4.6.4.2.4.3 Recommendation

It is recommended that common data elements be defined for a standard DoD software data system, using the USAF CPIN system as a guide. The common data elements could be identified and issued in a separate standard, or included in MIL-STD-482 to achieve DoD wide implementation.

4.6.4.3 Subpanel C - Security

The subpanel included:

George Kelly
George Smith
LTC Jim Harrington (first day)
Cal Showalter (Panel Co-Chair)

Issue: The proliferation of software intensive embedded computer resource based operational systems, the advances in micro chip memory technology and the emphasis on software language efficiencies have resulted in the increased potential for various kinds of security compromises and associated configuration management deficiencies that require policy attention by the JLC.

4.6.4.3.1 Scope: Security requirements demand that devices within or as elements of an embedded computer system be marked to reflect the status of that device/medium as to whether or not it contains classified data. Tapes must be marked and handled to reflect the highest level of classification contained within the entire tape. Fusible link programmable read only memory (ROM) must be marked (by etching, paint colors, etc.) to reflect the level of classification burned-in. Electrically erasable (EE) ROM's present a more difficult problem since the data which is loaded at one time may be classified and at a later time may be unclassified. Further an EEROM which has been removed from an equipment may have been only partially erased and may contain residual classified data.

Digital data which exists in any software medium is subject to reverse engineering which can yield derived sensitive data and this data may require security protection. The extent of compromise which can occur should be addressed in the context of both added complexity during employment and restrictions in overall fleet responsiveness if overclassification is imposed. The new generation of reprogrammable equipments in weapon systems was specifically designed to permit change in certain parameters. This capability to make rapid changes is new and may provide our armed forces with a combat edge. These on scene, combat adaptations could not have been accomplished in hard-wired systems of the past. These changes are often made by the insertion of software programmed into fusible link programmable ROMs. Frequently these parametric changes make use of highly classified enemy weapon system performance data.

As a related matter, the absolute needs of configuration management require that a PROM will be fully identifiable as to the host equipment, location, data status, etc. Using reverse engineering, it is possible to re-create portions of the source code from which the programmable ROM data was obtained. The source code reveals our weapon system performance and is normally classified, frequently at the secret level and the installed programmable ROM must be classified at the same level. The issue becomes "What is the classification of the programmable ROM after it has been programmed and identified for installation in the system when the programmable ROM is physically separated from that system?" The programmable ROM contains object code which yields source code which provides system performance information that has been classified. A further issue is "What is the classification of a programmable ROM which contained superseded threat data, after it has been removed from the system as a first step in accomplishing a threat data change or update?"

4.6.4.3.1.1 Discussion: The problem of essential use of classified information in the form of software as an integral part of the routine use of most of the equipments/systems in service inventory today is real. Procedures and safeguards for classified information, if not fully reflected in total system concepts, can begin to impede the operational flexibility of our forces and directly affect responsiveness and combat capability.

4.6.4.3.1.2 Conclusions: The problem warrants bringing together a multi-disciplined tri-service group to develop policy recommendations to govern the security and associated configuration management of operational software. Six preliminary guidelines or conclusions are capsulized here as a basis for tasking such a group:

a. All nonvolatile memory devices/elements can retain classified data even when removed from their host operational system, but the classification level generally cannot be determined by physical inspection. Electrically programmable memories (EAROM, EEPROM, Core, etc.) can be reprogrammed many times with classified data at any level, so that the highest level of intended classification seems to be the one of security significance.

b. Major weapon system operational programs may contain some parameters which are classified, but when those parameters are removed from the operational program and replaced with "dummy" values, the overall program becomes unclassified. This sanitized program permits simplified control and handling and greatly reduces potential for compromise.

c. Military equipments/systems containing software may be designed with integral, "permanent" data which incorporates classified information. When this is done the entire equipment/system, in the strictest sense, must be handled/safeguarded in accordance with procedures prescribed for the highest level of classified material it contains. It is possible to determine during the design phase the mix of data which an equipment will require in operation and to design the hardware such that the area which will contain the classified software data is well defined, partitioned, and segregated from the area which will operate with unclassified software data, maintenance actions, handling procedures, etc.. Such a design would provide a simpler means and an effective means for returning the equipment/system to an unclassified status after use.

d. It has been demonstrated that software in object code can be reverse engineered to source code which will re-create the level of classified performance information which a system manifests.

e. Since a system classification can be directly affected by the classification of the software loaded into it, and because there is normally no visible evidence of what software configuration is in memory, it might be concluded that the system must bear the classification of the highest level of software that is available for load. This conclusion should be modulated where the design of an automatic power down system sanitizes the software and provides verification of that sanitization event.

f. In all of these preliminary conclusions the paramount goal must be the deployment of systems which provide the maximum combat usefulness and flexibility of use consistent with the minimum security requirements which must be accommodated. Our military forces must be provided systems which can win despite essential security restrictions. Absolute security must not be an end within itself.

4.6.4.3.1.3 Recommendations:

It is recommended that a tri-service group having expertise in the areas of hardware design, software design and support, security, configuration management and operational employment of forces be commissioned to develop JLC policy recommendations and guidelines for the security aspects of current and future operational systems. In accordance with the preliminary conclusions above, the following candidate areas are offered for consideration and policy recommendations by such a follow-on tri-service group:

a. Classification of nonvolatile memory devices/elements.
Categories for consideration are: media as received from vendors, ready for issue, maintenance bench stores, installed and as removed from installation.

b. Marking of memory media. Categories for consideration same as in 1. above.

c. Configuration control policy

1. levels of S/W change to be reflected
2. master control vs local, on-site deployed control
3. master base-line change versus ephemeral change

d. Design/development policies to facilitate security requirements compliance

1. architecture/partitioning of software to isolate classified from nonclassified software regions
2. dummy load/sanitizing policy for classified software systems
3. crew insertion/removal of classified mission software on each mission basis

e. Program considerations relative to the above

1. increased cost in design/development to reduce operational complexity

2. increased cost if non-reuse policy of media is adopted to avoid risk of compromise thru storage and re-use of previously used media

f. Operational considerations due to the spectrum of increased complexity necessary to safeguard classified software media in daily operations.

4.6.4.4 Subpanel D - Configuration Status Accounting

The subpanel consisted of:

Mr. Rich Pariseau (Chair)
LTC Jim Harrington (second day)
Maj Ken Savage
LTC Bill Sanders
Mr. John Ruckstuhl

The major issue addressed by the configuration management (CM) panel was determining the scope of software CM (SCM). Among the targeted products of the CM panel was a multiservice definition of SCM.

Subpanel D was initially requested to address methods of tracking the configuration and the need for a central backup/repository for software. Specific initial areas of consideration included the hazards of single site storage, automated methods of tracking the configuration, physical control of the data base, and CM of the technical data as it relates to releases of the software. Suggested initial products for the subpanel included definition of the physical CM products, policy recommendation on the need for backup, identification of what must be transferred from the developer (government or contractor) to the post deployment software support (PDSS) agency, and ground rules for an automated tracking and status accounting system.

Following the first day's meetings and discussions, Subpanel D became aware that the other Subpanels of Panel F were already addressing the issues of defining the CM products (configuration identification), controlling those products (configuration control), and assuring the quality of those products (configuration auditing). Subpanel D decided, therefore, to concentrate on issues involving tracking of the CM products. In CM terminology, this area is called configuration status accounting (CSA).

The subpanel established its major issues to be the definition of CSA and the determination of its scope. The subpanel addressed five questions:

- a. What CSA information is stored, tracked, and reported?
- b. How should CSA information be stored, tracked, and reported?
- c. To whom should CSA information be available?
- d. Where should CSA information be stored?
- e. What developer contractual constraints are required?

The subpanel established the following definition for CSA:

CSA is the function which tracks, stores in a generic data base, and reports upon the results of configuration identification, configuration control, and configuration auditing.

In the following paragraphs it should be noted that storage of CSA information and tracking of CSA information are interrelated. To effectively track CSA information, it must be stored in some form of a data base. If there is no requirement to track CSA information, there is also no requirement to store it in a data base.

4.6.4.4.1 What CSA Information is Stored, Tracked, and Reported

From the point of view of the CSA process, there are three classes of information to be stored and tracked:

- a. The results of the configuration identification process.
- b. The results of the configuration control process.
- c. The results of the configuration auditing process.

It is emphasized that the CSA data base consists of information relating to the software product as opposed to the actual configuration items. The product software, its documentation, change proposals, minutes of meetings, and other pertinent information would be stored in other data bases. The CSA information that is reported consists of formatted excerpts from the CSA data base.

4.6.4.4.1.1 Information Stored and Tracked in the CSA Data Base

Results of the configuration identification process consists of the identification of all configuration items (CIs) comprising the product baseline configuration, including subsequent changes. Results of the configuration control and auditing processes consist of a record and audit trail of all significant CM events that occur from the moment that configuration control of a CI is assumed by the government. This includes information concerning:

- a. Trouble Reports (TRs).
- b. Change Requests (CRs).
- c. Document Discrepancy Reports (DDRs).
- d. Engineering Change Proposals (ECPs).
- e. Configuration Audit Results.
- f. Release Notices.
- g. Configuration Control Board (CCB) Meetings.

The collection of information in the CSA data base must be sufficient to describe the history, the present status, and anticipated configuration control events relating to the product baseline CIs.

4.6.4.4.1.2 Information Reported from the CSA Data Base

The subpanel identified five reports to be provided by the PDSS agency as standard CSA reports. These are described below.

4.6.4.4.1.2.1 CSA Data Base Top Level Summary

The purpose of this report is to provide a summary overview of the project baseline's configuration status. Context of the report includes:

- a. Identifying information (project, date, agency, location).
- b. Baseline information summary
 - (1) Identification information (release, users, location).
 - (2) Configuration identification (upper level CIs).
 - (3) Statistics on CIs.
- c. Change information summary.
 - (1) Statistics (number open, deferred, in process, closed).
 - (a) Provide the statistics for TRs, CRs, DMRs, and ECPs.
 - (b) Indicate the change in total number for each category since the last report.
- d. Historic Information.
 - (1) For the TRs, CRs, DMRs, and ECPs indicate the oldest and the most recent for the categories of open, deferred, in process, and closed.

4.6.4.4.1.2.2 CCB Minutes Summary

The purpose of this report is to provide a summary of a CCB meeting. Context of the report includes:

- a. Identifying information (date, location, purpose).
- b. List of official CCB actions taken.
- c. Information required to obtain copies of the complete CCB minutes.
- d. Anticipated date of the next CCB meeting.

4.6.4.4.1.2.3 CCB Minutes Summary Directory

The purpose of this report is to identify all available CCB Minutes Summary Reports. The context of the report includes a listing of the identifying information from all of the CCB Minutes Summary Reports.

4.6.4.4.1.2.4 Configuration Control Status Summary

The purpose of this report is to provide summary information concerning TRs, CRs, DMRs, and ECPs. The context of the report includes:

- a. Identification.
- b. Description (short form).
- c. Status (analysis, approved, implementation, testing, completed).
- d. Baseline impact.
 - (1) Affected CIs.
 - (2) Schedule for inclusion.

4.6.4.4.1.2.5 Configuration Item Summary

The purpose of this report is to provide summary information related to releases of the product. For each baseline product configuration, the context of the report includes:

- a. Identifying information (CI number, dates, release notice, users).
- b. Change information (previous baseline, ECPs incorporated).
- c. Location of product items and procedures for obtaining them.

4.6.4.4.1.3 Recommendation

The subpanel recommends that DoD establish the policy that all services store and track the same elements of CSA information. Additionally, the subpanel recommends that DoD establish the policy to standardize generically the information context of commonly used reports while allowing the flexibility of producing more specialized reports and data access to accommodate the needs of specific users of CSA information.

4.6.4.4.2 How should CSA Information be Stored, Tracked, and Reported

The subpanel concluded that all services should utilize a common, automated data base system to store, track, and report CSA information.

4.6.4.4.2.1 Storage and Tracking

The subpanel concluded that all services should utilize the same methods for storage and tracking of CSA information. This may be achieved through a sequence of steps. Initially, DoD should require that all PDSS agencies store and track the same elements of CSA information and utilize the same generic format for that information. For example, a common DoD configuration identification system should be used, dates should use a common format, and the information concerning CRs, TRs, DDRs, and ECPs should be identical across all PDSS agencies.

Initially, DoD should not restrict the methods utilized for storage and tracking. In some cases, manual methods will be sufficient for the amount of CSA information required to track the product. In other cases, data base systems of varying degrees of complexity will be necessary. The long range objective, however, should be to fully automate the storage and tracking methodology. In this automated system, changes to any CI should result in automatic generation of data for the CSA data base. A current example of this kind of automated storage and tracking of CSA data is the Naval Air Development Center's Facility for Automated Software Production (FASP).

4.6.4.4.2.2 Reporting

The subpanel concluded that with the exception of the handling of classified information, the common, automated CSA data base system should distribute all CSA reports via electronic communications. Receivers of electronically transmitted reports should have hard copy capability at their sites or should have the capability of requesting hard copy CSA information from the PDSS agency. CSA reports should be generated by the CSA agency in response to specific CM events, periodically, and in response to requests from interested parties.

4.6.4.4.2.3 Recommendation

The subpanel recommends that DoD develop a common, automated CSA data base system and require its use by all services. Analysis should be performed to establish the best approach (e.g., centralized, distributed) and

then that approach should be implemented. It is recommended that DoD immediately establish a common definition across all services for the elements of CSA information to be stored and tracked and for their formats. Initially, the medium of storage and tracking and the methods of entering data and updating the information in the CSA data base should reflect the capabilities of existing PDSS agencies. As DoD develops its automated CSA data base system, manual entry of data should be replaced by automated entry. Because the CSA activity is an activity that lends itself to electronic transmission of information, it is recommended that the developed CSA data base system utilize various transmission linkages that would be determined based upon the data requirements of the specific PDSS agency.

4.6.4.4.3 To Whom Should CSA Information be Available

CSA reports fall into two categories:

- a. Reports instigated by the PDSS agency responsible for CM.
- b. Reports requested by activities requiring CM information (sponsors, users, contractors, researchers).

4.6.4.4.3.1 Reports Instigated by the PDSS Agency

The CSA data base top level summary report should be provided to CCB members prior to CCB meetings. The CCB minutes summary report should be provided to all interested parties (CCB members, support agencies, users, sponsors) immediately following the CCB meeting. The configuration control status summary report should be provided periodically to all interested parties (users, support agencies). The period should be frequent enough to assure that the interested parties have a current understanding of ongoing PDSS events. The CI summary report should be provided to all interested parties (users, support agencies, sponsors) whenever a formal release of a baseline occurs.

4.6.4.4.3.2 Reports Requested by Activities

The subpanel concluded that the PDSS agency should pursue the objective of full disclosure of any CSA information to any interested requester. With the exception of classified material, all CSA information should be available to any authorized requester upon formal request from the PDSS agency.

4.6.4.4.3.3 Recommendation

It is recommended by the subpanel that DoD policy allow the widest dissemination of the CSA information. The DoD policy for controlling access to CSA information should be delegated to the government agency that controls the related CIs.

4.6.4.4.4 Where Should CSA Information be Stored?

The CSA data base should be conveniently accessible to the PDSS agency responsible for configuration control of the product. In the case of electronic transmission capability, it is not necessary for the data base to be physically located at the PDSS agency site. In the case of a manual implemented data base (written records, files), the data base should be located at the site of the PDSS agency.

The subpanel concluded that the CSA data base should be backed up in at least one location physically separate from the primary storage site. The backup data base should be periodically updated on a medium that will allow reestablishment of the primary data base in a reasonable time. At each time of creation, the backup data base must be certified identical to the primary data base.

4.6.4.4.4.1 Recommendation

The subpanel recommends that DoD establish a policy for creation of backup CSA data bases, certification of these data bases, and physical storage in at least one location physically separate from the primary storage site.

4.6.4.4.5 What Developer Contractual Constraints are Required?

The development contract must specify that the identification of the configuration items comprising the product baseline configuration and a record of significant configuration control events be provided to the government in a form compatible with the government established configuration status accounting data base (written format, machine readable, directly entered into a data base as pertinent). It is desirable that the developer provide configuration status accounting information to the government in a form compatible with the government established CSA data base rather than accept different developer's methods. Once DoD has developed a common, automated data base capability for CSA, it is recommended that the development contractor be required to utilize that capability in order to facilitate the transfer of CSA information to the government.

4.6.4.4.5.1 Recommendation

Upon development of a DoD common, automated CSA data base system, all contractors involved in major software developments should be required to utilize that system. Until the development is achieved, all contractors involved in major software developments should be required to provide CSA information in a form compatible with the government defined CSA data base.

4.6.4.4.6 Summary

The DoD should develop a common, automated CSA data base system. All agencies and contractors involved in major software development should be required to utilize that system. Information transfer in the system should be performed electronically. The CSA data should be backed up by a certified, redundant data base stored at a site physically separate from the PDSS agency site and created frequently enough to allow reestablishment of the CSA data base in a timely fashion. With the exception of classified data, the DoD should establish the policy of full disclosure of CSA information to all interested parties.

As an interim measure, the DoD should establish standard identification, format, and reporting of all elements of CSA information. All agencies and contractors involved in major software development should be required to utilize and/or deliver to the government CSA information in the prescribed format. Delivery should be contractually required in a format and medium compatible with the PDSS agency's interim CSA data base system.

4.6.5 Recommendations

4.6.5.1 Policy Recommendations

It is recommended that the JLC develop a PDSS CM policy document which requires that DoD/service directives, military standards and guidebooks relating to software management, acquisition and support contain the following PDSS CM considerations:

- a. Participation by the PDSS activity in the development phase to influence development phase configuration management practices and assure continuation of these practices into the deployment phase (see section 4.6.5.2(a) for implementation recommendation).
- b. A definition of software related products which must be developed by the development agency and turned over to the PDSS agency. These software related products are detailed in section 4.6.4.1.1 of this report.
- c. The development of a transition plan, jointly prepared by the development activity and the PDSS activity with the development activity having primary responsibility for generation of the plan. Minimum content requirements for the transition plan are contained in section 4.6.4.1.2.3 of this report.
- d. The establishment of a DoD wide requirement for a standard CPIN system. A proposed numbering system has been developed by the USAF and is contained in Appendix C.
- e. Computer Resources Life Cycle Management Plans must include provisions for handling multiple, parallel baselines.
- f. The establishment of a DoD wide requirement that all services store and track the essential designated elements of CSA information. Paragraph 4.6.5.2 describes the actions required to allow implementation of this policy (see implementation recommendation section 4.6.5.2(b), (c)).
- g. The requirement for storage of all CSA data bases in at least one location physically separate from the primary storage site.

4.6.5.2 Policy Implementation Recommendations

- a. The JLC direct the inclusion of the following paragraph in MIL-STD-SDS draft of August 1983:

The contractor shall specify PDSS parameters appropriate for the support of all the delivered and deployed software as part of the S/W Development Plan. These parameters, as a minimum, shall include the requirements for a support facility, work force mix, IV&V, cost of ownership, support environment and configuration

management with details for change processing. These assessments, by necessity, are considered iterative in nature, shall be reviewed at CSCI design reviews and provide the basis for a PDSS plan to be implemented not later than establishment of the product baseline. (see 4.6.5.1(a) above.)

- b. The JLC establish a tri-service group to address the implementation of the CSA system including at least the following:

- * identification of the essential designated elements which comprise the CSA.
- * standardize the information content of these elements.
- * trade analysis on centralized versus distributed approach to CSA.
- * determine optimum data transmission linkages.
- * investigate and recommend reporting formats and distribution.

- c. The JLC, based on the above, support the development of a common, automated CSA data base system for use by all services during development and PDSS. ((b) and (c) above support the policy of 4.6.5.1(f)).

- d. The JLC support the definition of CSCI as now included in Appendix XVII of MIL-STD-483, Notice-2.

4.6.5.3 Special Recommendations

Subpanel C, during its deliberations, determined that the software security issue during PDSS was greater in scope than could be resolved at this workshop and accordingly the following special recommendation is offered: A tri-service group having expertise in the areas of hardware design, software design and support, security, configuration management and operational employment of forces be commissioned on an urgent basis to develop JLC policy recommendations and guidelines for the security aspects of current and future operational systems. Items for consideration by this group are contained in section 4.6.4.3.1.3.

**GUIDE FOR SELECTING CONFIGURATION
ITEMS (CIs)**
MIL-STD-483 (USAF) 21 MARCH 1979

- SHALL BE USED IN THE CI SELECTION PROCESS WHENEVER IT OCCURS DURING THE LIFE CYCLE
 - DEFINITION—"AN AGGREGATION OF HARDWARE/SOFTWARE, OR ANY OF ITS DISCRETE PORTIONS, WHICH SATISFIES AN END USE FUNCTION . . . CIs ARE THOSE SPECIFICATIONS ITEMS WHOSE FUNCTIONS AND PERFORMANCE PARAMETERS MUST BE DEFINED AND CONTROLLED TO ACHIEVE THE OVERALL END USE FUNCTION AND PERFORMANCE".
 - SELECTION—NORMALLY A FUNCTION OF ANTICIPATED DESIGN AND SHOULD BE INDEPENDENT OF THE CONCEPT FOR FUTURE PROCUREMENT.
- ITEMS TO BE MANAGED AS CIs SHOULD BE DETERMINED BY THE NEED OF THE GOVERNMENT TO CONTROL AN ITEM'S INHERENT CHARACTERISTICS OR TO CONTROL THAT ITEM'S INTERFACE WITH OTHER ITEMS. SELECTION IS A MANAGEMENT DECISION NORMALLY ACCOMPLISHED THROUGH THE SYSTEM ENGINEERING PROCESS IN CONJUNCTION WITH CONFIGURATION MANAGEMENT AND WITH THE PARTICIPATION OF LOGISTICS.

COMPUTER PROGRAM CI (CPCI) CONSIDERATIONS

- DECISION MADE BY THE DEVELOPER BASED UPON SYSTEM TRADE-OFFS AND THE NATURAL DECOMPOSITION OF THE SOFTWARE
- GENERALLY, THE EXECUTIVE/SUPERVISOR, FUNCTIONAL/APPLICATIONS, INPUT/OUTPUT, TEST AND SUPPORT PROGRAMS SHOULD BE INDIVIDUAL CPCIs. CPs WITH POTENTIAL USE IN MULTIPLE SYSTEMS SHOULD BE:
 - ▲ HIGHLY INTERRELATED CPs . . .
 - ▲ ESTABLISHED TO THEIR LARGEST FUNCTIONAL ELEMENT ASSEMBLIES OF CP ELEMENTS TO BE ACQUIRED FROM A SINGLE CONTRACTOR
- CPs TO BE DESIGNATED FOR OPERATION IN DIFFERENT MODELS OF COMPUTERS SHOULD BE SEPARATE CPCIs
- CPCIs SCHEDULED FOR DEVELOPMENT, TESTING, AND DELIVERY AT DIFFERENT TIMES MAY BE SEPARATE CPCIs IF THERE ARE DIFFERENT CONFIGURATIONS DUE TO DIFFERENT ADAPTATION DATA FOR EACH OPERATING LOCATION.

COMPUTER PROGRAM CI (CPCI) CONSIDERATIONS (CONT)

- **EFFECTS OF CI SELECTION**
 - ▲ CI SELECTION AFFECTS COST, SCHEDULE AND/OR PERFORMANCE FOR THE GOVERNMENT, PRIME CONTRACTORS, SUBCONTRACTORS AND SUPPLIERS.
 - ▲ SELECTION OF AN ITEM AS A CI FOR MANAGEABILITY MAY BE BASED ON ITS ADMINISTRATIVE COMPLEXITY, TECHNICAL (ENGINEERING) CRITICALITY OR MAINTENANCE (LOGISTICS) CRITICALITY.
 - TOO MANY CIs MAY RESULT IN EFFECTS HAMPERING VISIBILITY AND MANAGEMENT RATHER THAN IMPROVING IT.
 - TOO FEW CIs MAY RESULT IN COSTLY LOGISTICS AND MAINTENANCE DIFFICULTIES.

SA A BANDERAS
ANALYSTS INC

Attachment 4-6-1-3

V WO MM, RM

APPENDIX A
PARTICIPANTS

<u>CO-CHAIRMAN:</u>	Showalter, Mr. C. (Cal) Naval Air Systems Command (AIR-543C) Room 620, JP-2 Washington, DC 20361	(202) 746-0650 A/V 286-0650
<u>CO-CHAIRMAN:</u>	Schuman, Ms. A. (Toni) TRW Systems Group 1 Space Park, Bldg. 134, Rm. 6079 Redondo Beach, CA 90278	(213) 217-4079
<u>MEMBERS:</u>		
A	Savage, MAJ K. (Ken) U.S. Army Computer Systems Command (ACSC-TE) Ft. Belvoir, VA 22060	(703) 664-4878 A/V 354-4878
A	Gonzalez, Mr. L. (Leo) HQ USAADASCH (AT7A-CDS-R) Ft. Bliss, TX 79916	(915) 568-1056/2810 A/V 978-1056/2810
A	Taupeka, Mr. H. (Norm) USA CECOM (DRSEL-TCS-ED) Ft. Monmouth, NJ 07703	(201) 532-2319 A/V 992-2319
N	Anderson, Mr. G. (Gerry) Naval Ocean Systems Center (9133) San Diego, CA 92152	(619) 225-7615 A/V 933-7615
N	Pariseau, Mr. P. J. (Rich) Naval Coastal Systems Center (310) Panama City, FL 32407	(904) 234-4113 A/V 436-4113
N	Smith, Mr. G. (George) Pacific Missile Test Center (4020) NAS Pt. Mugu, CA 93042	(805) 982-8981 A/V 351-8981
MC	Sanders, LTCOL B. C. (Bill) USMC HQ USMC (LMC-3) Washington, DC 20380	(202) 695-5170 A/V 225-5170
AFLC	Wagoner, Mr. J. (James) OC-ALC/MMEDUC Tinker AFB, OK 73145	(405) 734-2227/5355 A/V 735-2227/5355
AFLC	LaVergne, Mr. R. (Bob) SM-ALC/MMIR (AFLC) McClellan AFB, CA 95652	(916) 643-3154 A/V 633-3154

AFSC	Faranello, Mr. J. (Joe) ESD/ALEQ Hanscom AFB, MA 01731	(617) 861-2922 A/V 478-2922
AFSC	Rhoades, LTC D. (Dusty) USAF PSMC/SE-T Ft. Belvoir, VA 22060	(703) 664-3477 A/V 354-3477
AFSC	Mayhan, Mr. A. (Al) AFDSNC/SC Gunter AFS, AL 36114	(205) 279-4462 A/V 446-4462
A	Soskins, Ms. F. (Frances) TELOS Computing 3420 Ocean Park Blvd., Suite 3050 Santa Monica, CA 90405	(213) 450-2424
A	Eichert, Mr. L. (Lew) Teledyne Brown Engineering 788 Shrewsbury Avenue Tinton Falls, NJ 07727	(201) 741-5008
N	Kelly, Mr. G. (George) Computer Science Corporation 6521 Arlington Blvd. Falls Church, VA 22046	(703) 237-1333 X341
N	Maibor, Mr. D. (David) Dynamics Research Corporation 60 Concord St. Wilmington, MA 01887	(617) 652-6100
N	Fumia, Ms. D. F. (Dreama) Veda Inc. 1755 S. Jeff Davis Hwy., Suite 700 Arlington, VA 22202	(703) 979-2700
AFLC	Bushey, Mr. M. (Max) Boeing Military Airplane Company 3801 South Oliver (M/S K31-26) Wichita, KS 67210	(316) 526-4571/3415
AFSC	Berlack, Mr. R. (Ron) Sanders Associates, Inc. 95 Canal St. (M/S NCA 1-4222) Nashua, NH 03061	(603) 885-5170
NTEC	Ruckstuhl, Mr. J. (John) Naval Training Equipment Ctr (N-401) Orlando, FL 32813	(305) 273-4891 A/V 791-4891

APPENDIX B
BIBLIOGRAPHY

1. MIL-STD-SDS "Defense System Software Development," 30 July 1983, and associated Data Item Descriptions
2. MIL-STD-483 (USAF) "Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs" (old and JLC versions)
3. MIL-STD-1679 (Navy) "Weapon System Software Development," 1 December 1978
4. MIL-STD-490 "Specification Practices" (old and JLC versions)
5. Final Report of the Joint Logistics Commanders Software Workshop (Monterey I), 1 October 1979.
6. Final Report of the Joint Logistics Commanders Software Workshop (Monterey II), 1 November 1981.
7. MIL-STD-480A "Configuration Control - Engineering Changes, Deviations and Waivers," 12 April 1978
8. DoD Directive 5000.29 "Management of Computer Resources in Major Defense Systems," 26 April 1976
9. DoD Directive 5010.19 "Configuration Management," 1 May 1979
10. SECNAVINST 5200 "Management of Computer Resources in Department of Navy System," (undated)
11. NAVELEXINST 5200.23 "Computer Software Life Cycle Management Guide," 1 March 1979
12. AFLC Reg. 800-21 "Management and Support Procedures for Computer Resources Used in Defense Systems," 21 January 1983
13. DARCOM Reg. 70-16 "Management of Computer Resources in Battlefield Automated Systems," 16 July 1979
14. NAVAIRINST 5230.9 "Policy and Procedures for the Establishment and Operation of Naval Air Systems Command Systems Software Support Activities," 14 June 1983

APPENDIX B: BIBLIOGRAPHY (CONTINUED)

15. AR 380-380 "Automated Systems Security,"
 14 October 1977
16. Note to CRMP Preparers, CECOM Publication (undated)
17. DoD Joint Services Configuration Management Regulation,
 1 July 1974. Implementing Instructions are: AR 70-37,
 NAVMATINST 4130.1A, MCO 4130.1A, AFR 65-3, DSAR 8250.4,
 NSA/CSS 80-14, DCAC 100-50-2 and DNA INST 5010.18

APPENDIX C
BRIEFING CHARTS

(PUBS COMMITTEE HAS THE ONLY COPIES)

**BASELINED SOFTWARE
CONFIGURATION MANAGEMENT:
AN AUTOMATED APPROACH**



**NAVAL TRAINING EQUIPMENT CENTER
SOFTWARE SUPPORT FACILITY**

1661

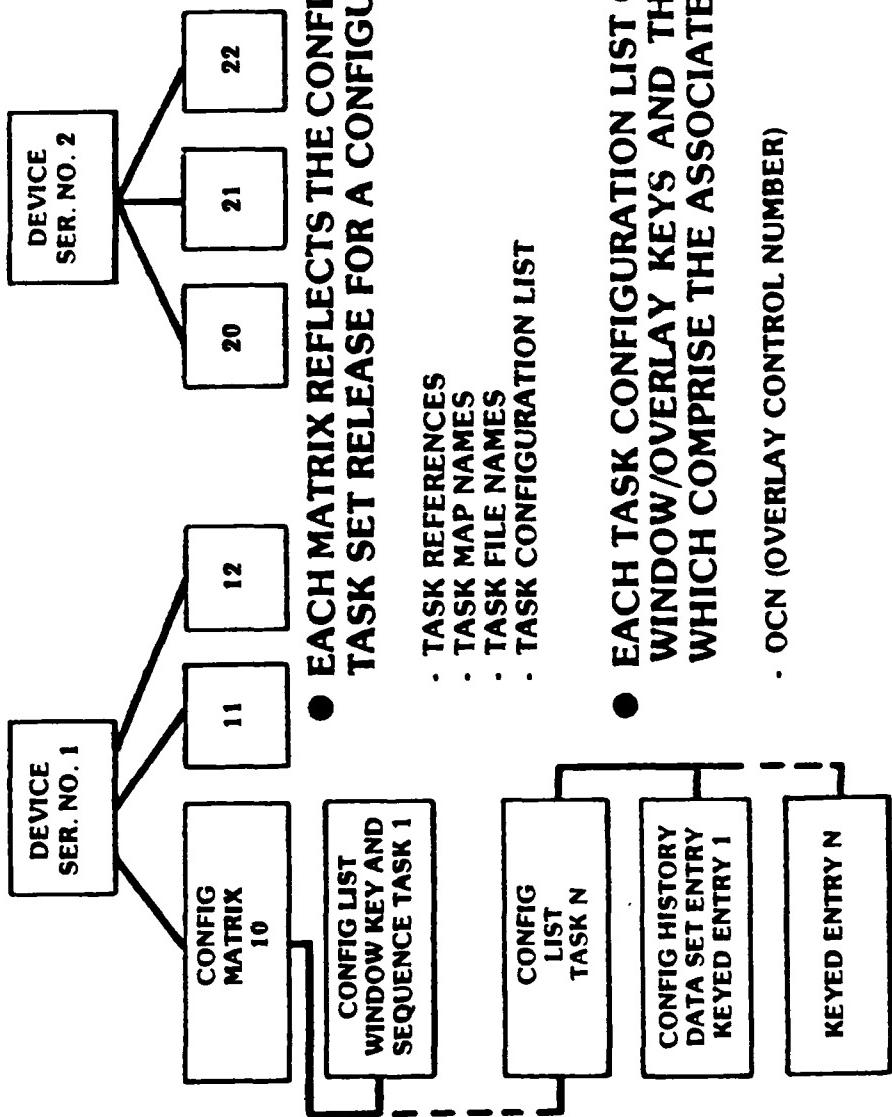
WHAT GOES WHERE??

- **BASIC SOFTWARE CONFIGURATION SERVICES:**

- PROTECT INTEGRITY OF SOFTWARE BASELINES.
- PROVIDE FOR EASE OF MODIFICATION TO SOFTWARE BASELINES.
- ESTABLISH AND MAINTAIN CONFIGURATION MODIFICATION HISTORY/AUDIT TRAILS.
- PROVIDE ACCESS SERVICES FOR BASELINE SOURCE, LISTINGS, AND APPLICABLE DATA

1864

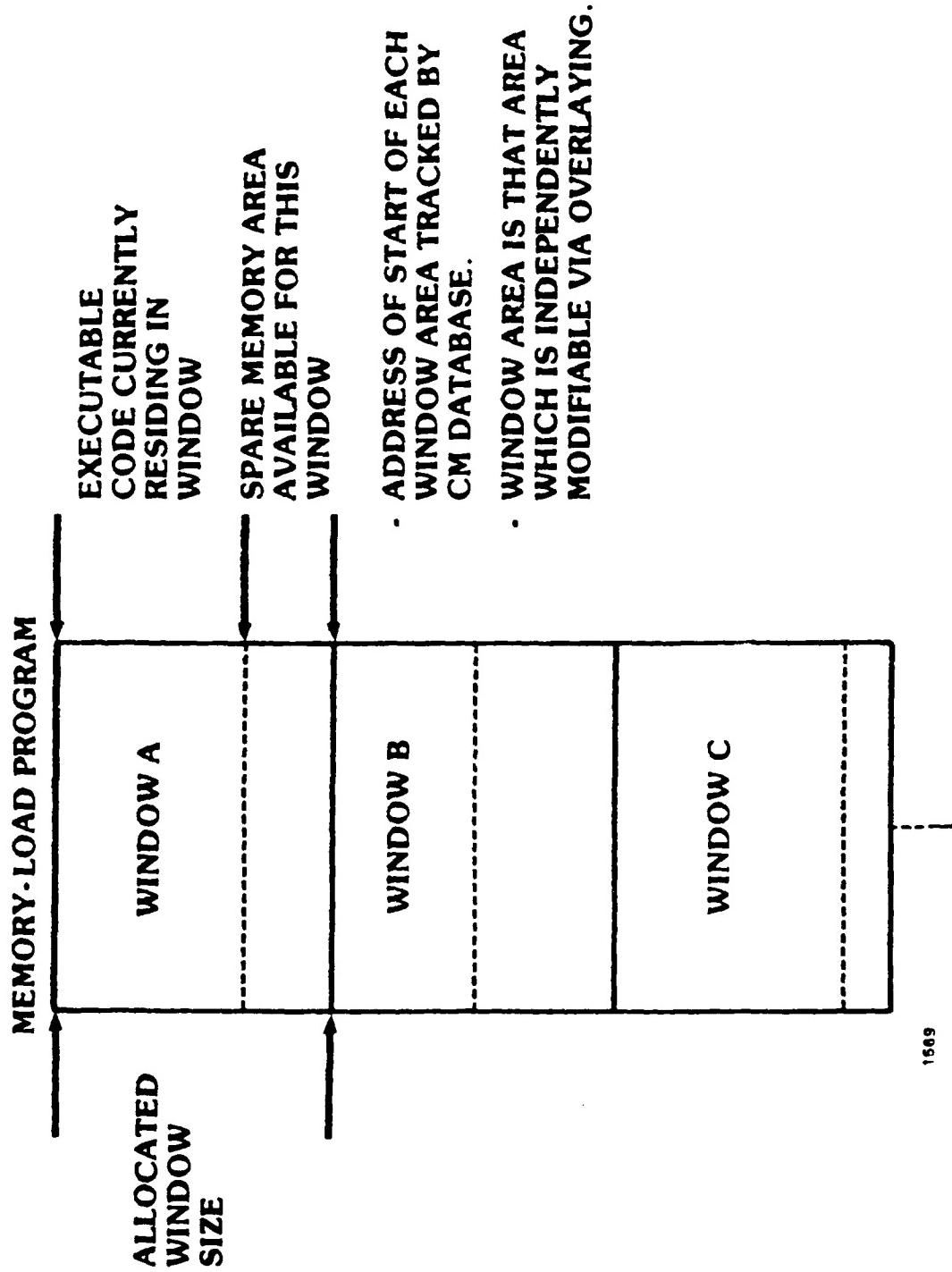
CONFIGURATION STRUCTURE



CONFIGURED TASK COMPONENTS

- **WINDOW:**
THAT SEGMENT OR AREA OF A CONFIGURED TASK IMAGE INTO WHICH EXECUTABLE CODE IS LINKED AT TASK GENERATION TIME OR OVERLAYERD AT LOAD TIME WHEN AN OVERLAY FILE IS LOADED.
- **OVERLAY:**
A STAND-ALONE FILE CONTAINING EXECUTABLE CODE WHICH HAS BEEN LINKED TO THE ADDRESS OF THE WINDOW INTO WHICH IT WILL BE LOADED DURING TASK LOAD INITIALIZATION.
- **ROOT:**
THAT SEGMENT OF A CONFIGURED TASK IMAGE WHICH IS CLASSIFIED AS PERMANENT AND NOT OVERLAYABLE. THIS IS THE MAIN SEGMENT OF THE LOADABLE TASK IMAGE TO WHICH THE FIRST OVERLAY OF THE OVERLAY CHAIN IS ATTACHED.

SEGMENTED TASK STRUCTURE (MIL-STD 1644A)



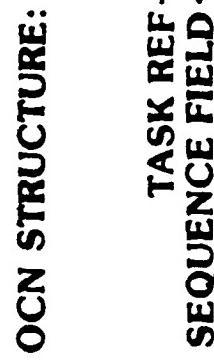
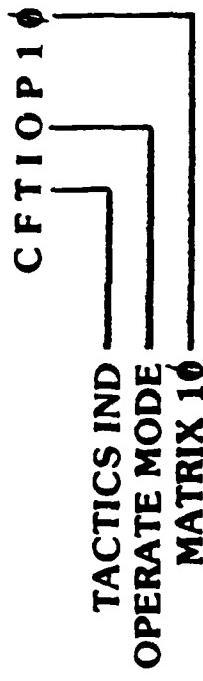
1669

OVERLAY HISTORY DATA SET

● 2235	MSCORE.V26 @2M. SCORE @2 XEWSKR SCOREND	31 JUL 80 .M2@ B2@ .F26 B26	NTEC SMF ENDPART	2253 0058@ ENDC \$4230@ \$42344 \$46EB8
● 2253	MSCORE.V28 @2 M. SCORE @2 XEWSKR SCOREND	09/02/81 .M2@ B2@ .F27 B27	NTEC SMF CHG ENDPART	2251 0058@ 2235 ENDC \$4230@ \$42344 \$46F1@
● 2251	MSCORE.29 @2 M. SCORE @2 XEWSKR SCOREND	01/20/82 .M2@ B2@ .F28 B28	NTEC SMF CHG ENDPART	2235 0058@ ENDC \$4230@ \$42344 \$46AE@

CONFIGURATION LIST

1040
1241
1042
1043
1160
1221
1036
1216
1211
1012
1226
1199
1023
1039
1202
1220
1128
1126
1122
1031
1214
1073
1200
1225
1112
1165



- THIS OCN SEQUENCE REPRESENTS THE CONSTRUCT OF TASK TKTIOPI1.
- THESE OCN's ARE KEYED TO WINDOW/OVERLAY CONTENTS FOR THIS PARTICULAR TASK CONFIGURED BY MATRIX 1.
- THESE OCN's ARE 'INSTALLED' (IN) FOR MATRIX 1.

OCN STATUS DATA SET

WINDOW/OVERLAY LIFE CYCLE STATUS

STATUS ENTRY: MATRIX ID + STATUS + DATE

1Φ	PN	YYMMDD
1Φ	UT	YYMMDD
1Φ	CK	YYMMDD
2Φ	CK	YYMMDD
1Φ	AP	YYMMDD
2Φ	AP	YYMMDD

TYPICAL STATUS SEQUENCE FOR A SINGLE OCN:

PN->UT->CK->AP->IN (MATRIX + 1)->RP (MATRIX + 2)
RJ RJ RJ RM

PN - CREATION PENDING

UT - UNTESTED

CK - CHECK-OUT

AP - APPROVED (CEMENTED)

IN - INSTALLED (TASK IMAGE)

RP - REPLACED

RM - REMOVED (WITHOUT REPLACEMENT)
RJ - REJECTED

WINDOW CONTENTS MODIFICATION

- PARENT/SIBLING RELATIONSHIPS

- AN OVERLAY IS CREATED (SIBLING) BY REFERENCED CHANGES TO AN EXISTING SEED OVERLAY (PARENT)

- THE PARENT OVERLAY IS ALWAYS THE INSTALLED WINDOW

CFTIOP 1#	PARENT WINDOW	1#42	MIEXEC. VIA	.M21	B21	1#54 END C
1#40	MIEXEC	EXEC	FYA	BIA		
1241		EXEC		CZ1	B21	
1#42	EXECOVLY			CZ2	B22	
1#43	EXECSKED			A6TIME	B22	
OVY 1#42	A6TIME	EOFMOD		EOFMOD	BZ3	ENDPART
CHG EXEC, FYA, FYB		EXECIEND				
CHG EOFMOD, CZ3, FZO						
1#54	MIEXEC. VIB					ENDC 1#42
	MIEXEC	EXEC		.M21	B21	
		EXEC		FYB	B1B	CHG
	EXECOVLY			CZ1	B21	
	EXECSKED			CZ2	B22	
	A6TIME			A6TIME	BZ2	
	EOFMOD			EOFMOD	FZO	CHG
	EXECIEND					ENDPART

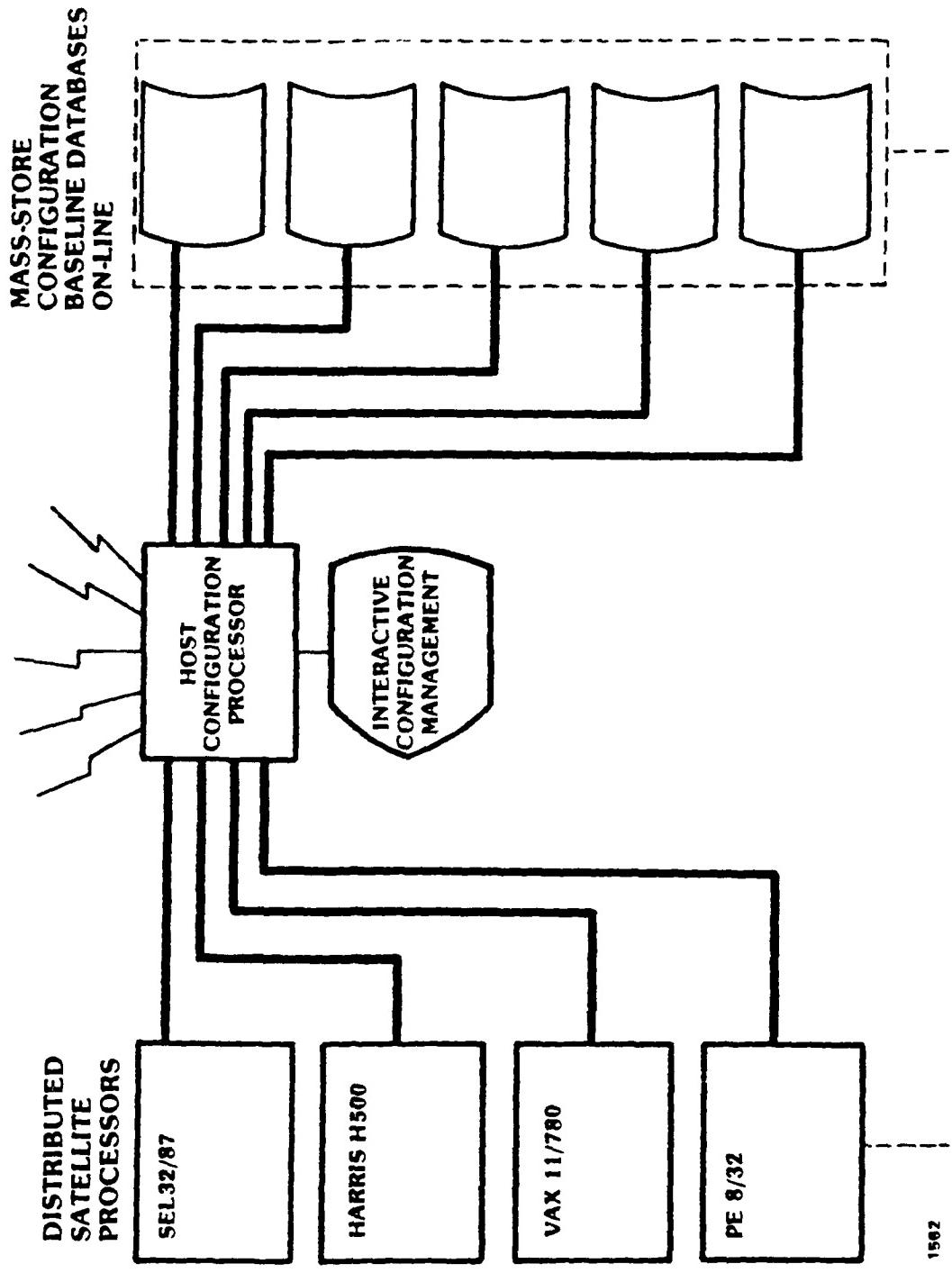
CONFIGURATION FREEZE CYCLE

MATRIX 11 CFAFOP11		INSTALLED	APPROVED	INSTALLED
C149			C149	C253
C253				C145
C145				C146
C146				C154
C154				C228
C228				C252
C239	C252			MODIFICATION PROCESS CONTINUES FROM HERE
C236	C147			
C219	-			C147
C237	C268			-
				C268
C188	C266			C266
C178				C188
C177				C178
C152				C177
C216				C152
C214				C243
C249				C214
C234				C249
C179				C234
				C179

1865

DISTRIBUTED NETWORK AT SSF FOR SOFTWARE CONFIGURATION CONTROL AND PROGRAM PROCESSING

MULTI-USER INTERFACE



1502

MODULE-IN-WORK MANAGER

MANAGEMENT DATA DISPLAYS	OPEN	CLOSED
- MODIFICATION CONTROL NUMBER	(#5)	(#6)
- COGNIZANT ENGINEER RELATIONS	(#7)	(#8)
- PARENT OCN RELATIONS	(#9)	(#10)
- SOURCE MODULE RELATIONS	(#11)	(#12)
- OVERLAY MODULE RELATIONS	(#13)	(#14)
- TASK LEVEL RELATIONS	(#15)	(#16)
- ALL OPEN ASSIGNMENTS	(#17)	
- TERMINATE (99)		
- ENTER FUNCTION SELECTION (XX)		

>

1668

USAF

COMPUTER PROGRAM IDENTIFICATION NUMBER
(CPIN) SYSTEM

JOINT LOGISTICS COMMANDERS
JOINT INDUSTRY AND GOVERNMENT SOFTWARE WORKSHOP

ORLANDO I



C-15

JAMES A. WAGONER
CPIN SYSTEM MANAGER
OKLAHOMA CITY ALC/MMEDUC
31 OCT - 4 NOV 1983

USAF
COMPUTER PROGRAM
IDENTIFICATION NUMBER
(CPIN) SYSTEM

WHAT IS CPIN

- CENTRALIZED DATA SYSTEM
- STANDARDIZED IDENTIFICATION
- CONSOLIDATED INDEX (COMPENDIUM)
- MAINTAIN USER REQUIREMENTS
- CONTROLLED DISTRIBUTION
- MANAGEMENT VISIBILITY

BACKGROUND

- WHY A CPIN SYSTEM FOR ECS SOFTWARE?
 - FORMERLY MANAGED AS HARDWARE OR TECHNICAL DATA
 - NO STANDARDIZED IDENTIFICATION OF SOFTWARE
 - SOFTWARE NOT INDEXED OR ANNOUNCED
 - REQUIREMENTS FOR SOFTWARE NOT MAINTAINED
 - INSUFFICIENT DISTRIBUTION OF SOFTWARE
 - INCOMPLETE FOLLOW-ON OR UPDATE DISTRIBUTION
 - PARTIAL OR SPORADIC MANAGEMENT REPORTS

OBJECTIVE

- **PROVIDE SUPPORT FOR THE EFFECTIVE, ECONOMICAL,
EFFICIENT MANAGEMENT OF EMBEDDED COMPUTER SYSTEM
(ECS) SOFTWARE AND TIMELY SUPPORT OF THE ECS USERS**

CPIN SYSTEM CONCEPT

- APPLY PRINCIPLES OF HARDWARE CONFIGURATION MANAGEMENT TO ECS SOFTWARE
- IDENTIFY ECS SOFTWARE AS CONFIGURATION ITEMS (CPCIs)
- IDENTIFY EACH CPCI WITH A CPIN
- CONTROL THE ASSIGNMENTS OF CPINS AND OTHER CPIN SYSTEM OBJECTIVES THROUGH A CENTRALIZED DATA SYSTEM

COMPUTER PROGRAM CONFIGURATION ITEM (CPCI)

- SOFTWARE ACQUIRED, DEVELOPED, MANAGED OR USED UNDER THE CONFIGURATION MANAGEMENT CONCEPT
- GENERALLY REFERRED TO AS EMBEDDED COMPUTER SYSTEM (ECS) SOFTWARE
- INCLUDES COMPUTER PROGRAMS AND ENGINEERING DOCUMENTATION

CPCI DOCUMENTATION

- CLASSIFIED IN TWO CATEGORIES
 - ENGINEERING DOCUMENTATION
 - USER DOCUMENTATION

ENGINEERING DOCUMENTATION

- USED TO DEVELOP A CPCI OR TO DEVELOP CHANGES
TO A CPCI
- ESTABLISHES THE CONFIGURATION BASELINE FOR A CPCI
- ENGINEERING DOCUMENTATION PACKAGE INCLUDES:
 - SPECIFICATIONS
 - ENGINEERING DESIGN/TEST/INTERFACE DATA
 - TEST PLANS
 - INTERFACE CONTROL DRAWINGS
 - FLOW DIAGRAMS
 - LISTINGS
 - SOURCE DATA

USER DOCUMENTATION

- **PURPOSE:**
 - OPERATIONAL USE
 - EQUIPMENT CHECKOUT
 - EQUIPMENT INSTALLATION
 - TROUBLE SHOOTING
 - LOADING THE CPCI

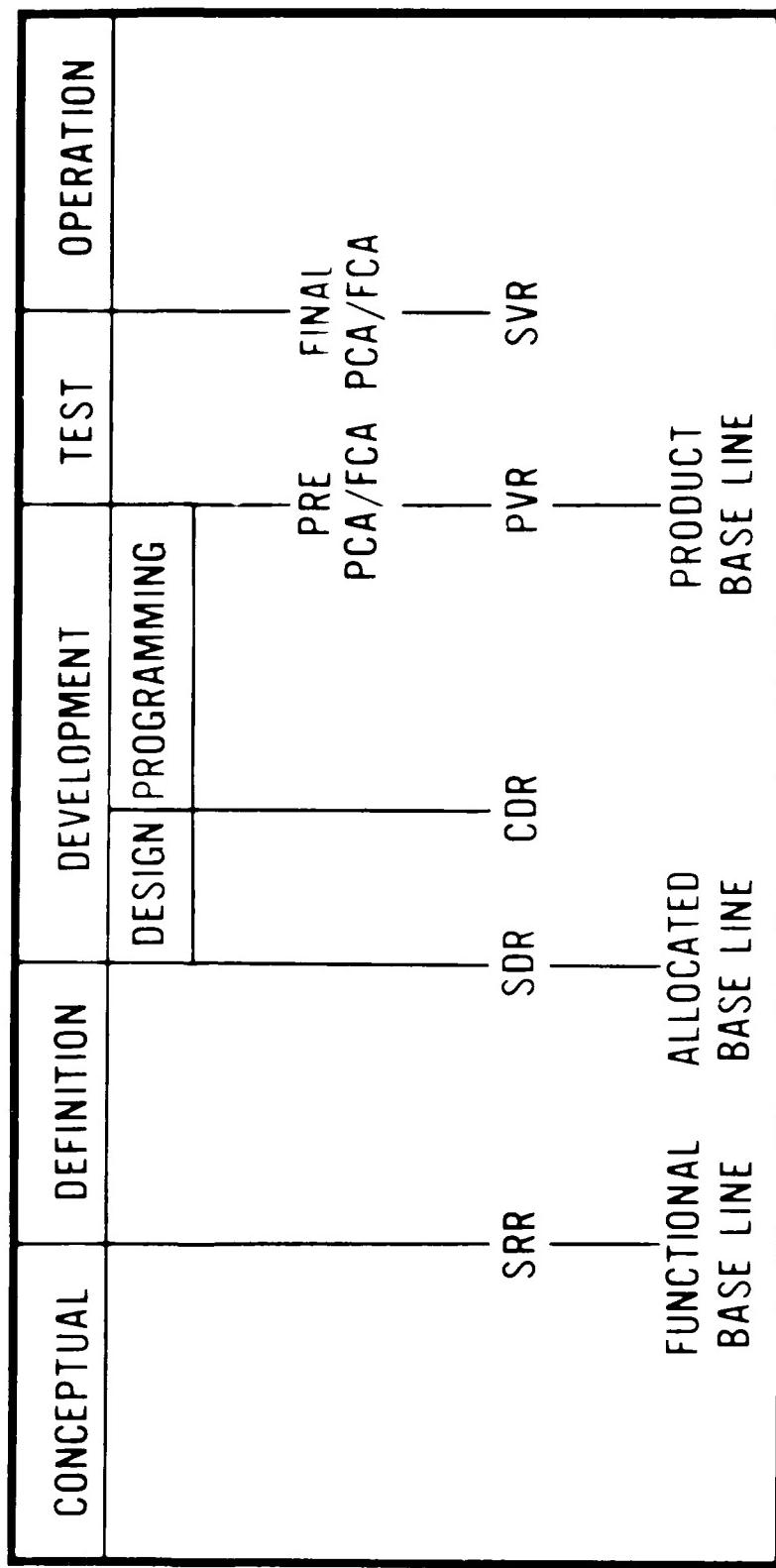
- **TYPES:**
 - COMPUTER OPERATORS MANUALS
 - USER MANUALS
 - TEST EQUIPMENT MANUALS
 - TEST PROCEDURES MANUALS
 - USER MAINTENANCE MANUALS
 - REFERENCE MANUALS

CPIN ASSIGNMENT

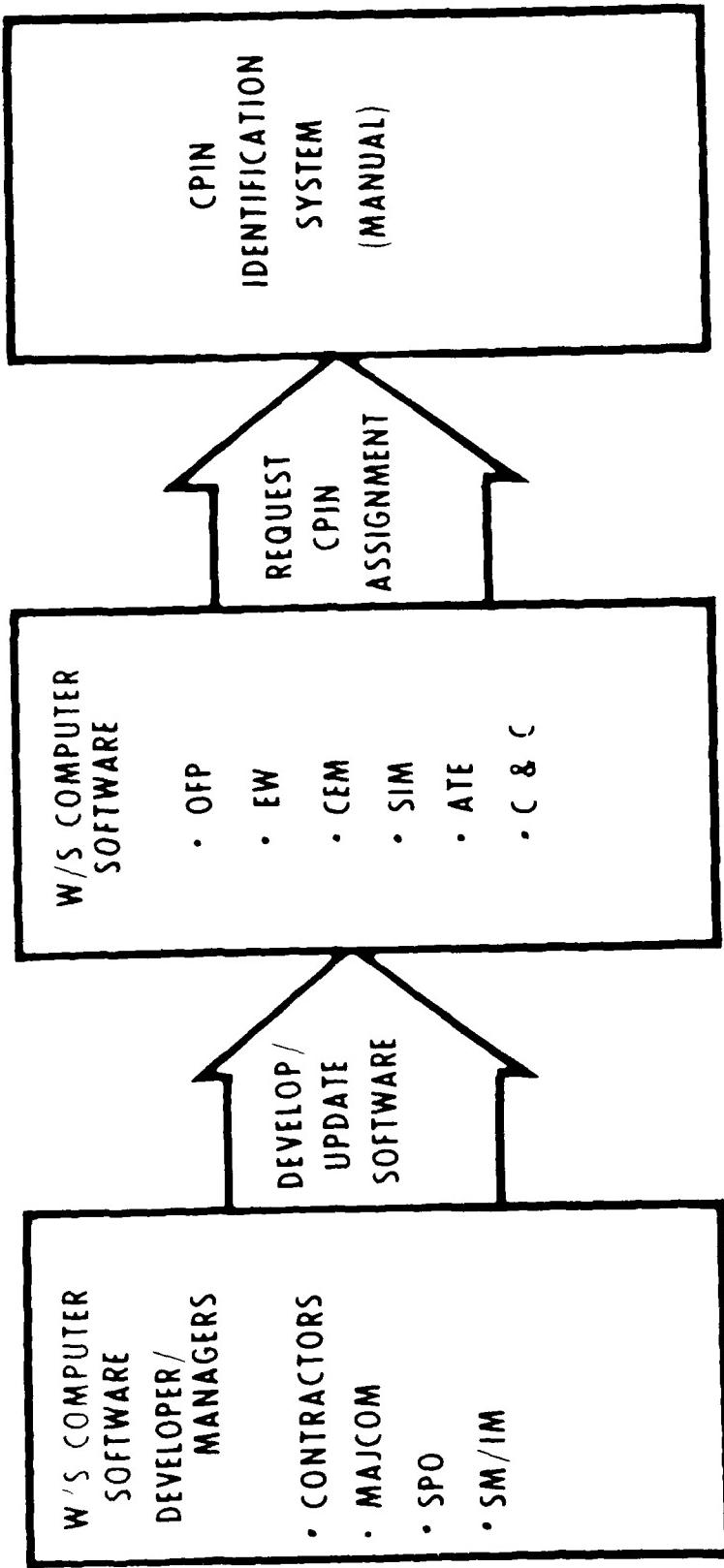
COMPUTER PROGRAM IDENTIFICATION NUMBERS (CPINs)

- A STANDARDIZED, SIGHT RECOGNIZABLE, IDENTIFIER ASSIGNED TO EACH CPCI AND ASSOCIATED ENGINEERING PACKAGE
- ASSIGNED AS SOON AS THE CPCI HAS BEEN DESIGNATED
- PRIOR TO TURNOVER OR PMRT

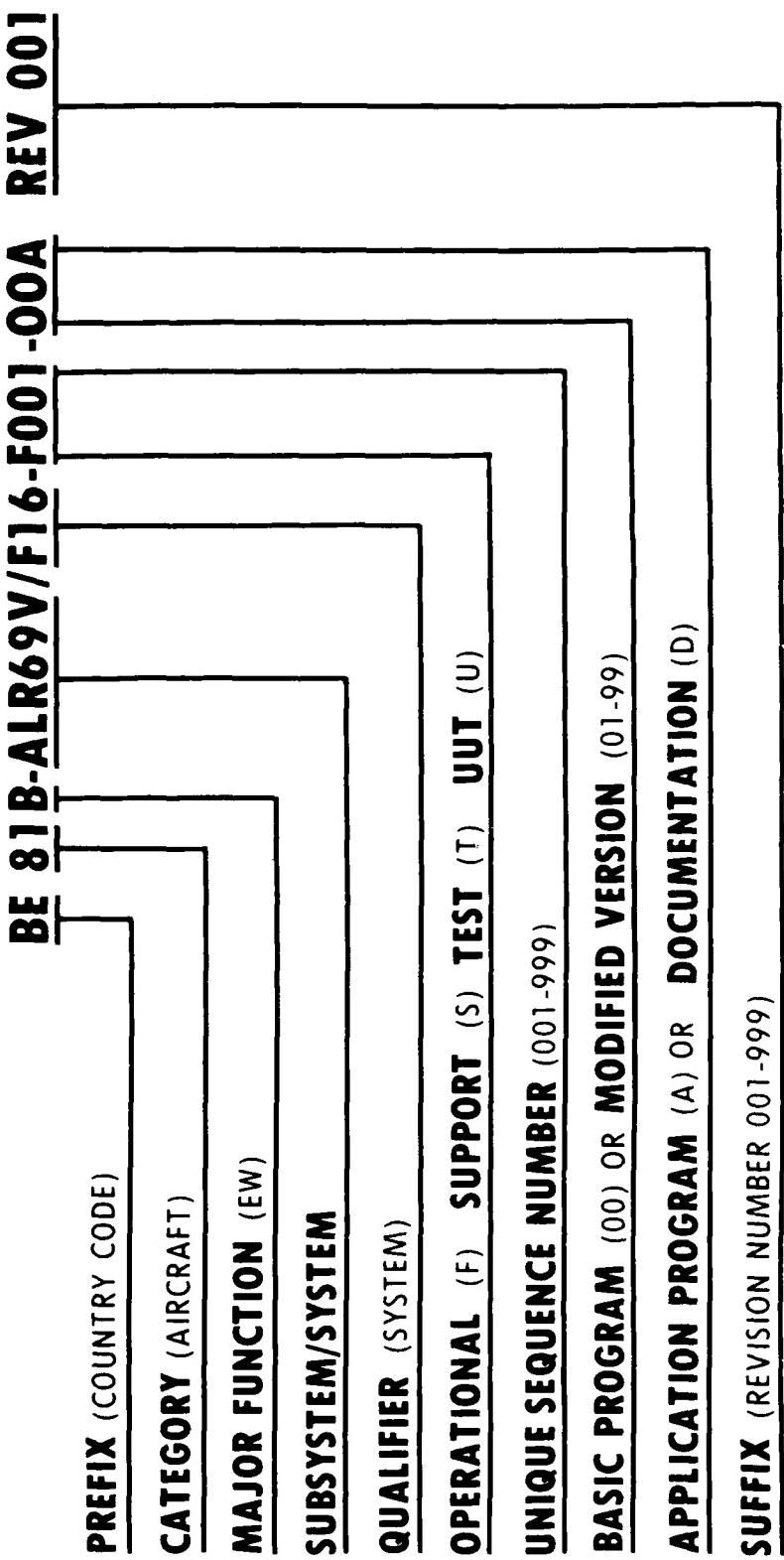
ADS PROJECT MANAGEMENT



CPIN IDENTIFICATION



IDENTIFICATION PATTERN



CPIN PREFIX

COUNTRY CODE

NE 81B-ALQ131V/F16A-U001-00A REV 001
—

- ASSIGNED TO COUNTRY STANDARD SOFTWARE
- NOT ASSIGNED TO CONSORTIUM SOFTWARE

CONSORTIUM AND COUNTRY STANDARD SOFTWARE

- SOFTWARE USED BY OTHER COUNTRIES
- CONSORTIUM SOFTWARE
 - USED JOINTLY BY THE USAF AND ANOTHER COUNTRY
 - DEVELOPED BY THE USAF - SOLD TO ANOTHER COUNTRY
 - DEVELOPED AND OWNED JOINTLY BY THE USAF AND ANOTHER COUNTRY
- COUNTRY STANDARD SOFTWARE
 - NOT USED BY THE USAF
 - DEVELOPED AND OWNED BY ANOTHER COUNTRY
 - IDENTIFIED IN THE CPIN SYSTEM AND SUPPORTED BY THE USAF

CPIN CATEGORIES

NE 81B-ALQ131V/F16A-U001-00A REV 001

<u>CATEGORY</u>	<u>TITLE</u>
81	AIRCRAFT
82	MISSILES
83	GROUND COMMUNICATION-ELECTRONICS
84	SIMULATORS OR TRAINERS
85	TEST STATIONS OR TESTORS
87	GENERAL PURPOSE COMPUTERS
88	OTHER
89	SPACE VEHICLES AND SUPPORT SOFTWARE
91	COMMAND AND CONTROL

CPIN MAJOR FUNCTIONS

NE 81B-ALQ131V/F16A-U001-00A REV 001

- A—OPERATIONAL FLIGHT
- B—ELECTRONIC WARFARE
- C—COMMUNICATIONS
- D—DATA PROCESSING OR DISPLAY
- E—ENGINES
- F—FLIGHT CONTROLS
- G—GUIDANCE
- H—NAVIGATION
- J—WEAPONS DELIVERY
- K—FIRE CONTROL
- L—MISSILE LAUNCH
- M—METEOROLOGY
- N—ENVIRONMENTAL AND EGRESS
- P—PHOTOGRAPHY
- Q—ELECTRONIC AND ELECTRICAL
- R—ARMAMENT AND MUNITIONS
- S—FUEL
- T—MULTIPLE MAJOR FUNCTIONS
- U—HYDRAULIC, PNEUMATIC,
PNEUDRAULIC, VACUUM
- V—GENERAL PURPOSE OR SUPPORTIVE
- W—SURVEILLANCE, TRACKING, IFF
- X—RESERVED
- Y—RESERVED
- Z—OTHER

SYSTEM OR SUBSYSTEM

NE 81B-ALQ131V/F16A-U001-00A REV 001

- SUBSYSTEM
- SYSTEM
- "AN" NOMENCLATURED
- ACRONYMS
- ABBREVIATIONS
- CONTRACTOR PART NUMBERS
- QUALIFIERS

TYPE SOFTWARE AND SEQUENCE NUMBER

NE 81B-A1Q131V/F16A-U 001-00A REV 001

- **TYPES OF SOFTWARE**

F — OPERATIONAL
S — SUPPORT
T — IN-PLACE OR SELF TEST
U — UNIT UNDER TEST
C — COMBINATION
D — MASTER

- **SEQUENCE NUMBER**

COMBINATION AND MASTER CPIN

- COMBINATION CPIN

NE 81B-ALQ131V/F16A-C000-00A REV 001

- MASTER CPIN

CPCI MASTER CPIN

NE 81B-ALQ131V/F16A-D000-00A REV 001

DOCUMENTATION MASTER CPIN

NE 81B-ALQ131V/F16A D000-00D REV 001

BASELINE SOFTWARE OR VERSION PROGRAM OR DOCUMENTATION

NE 81B-ALQ131V/F16A-U001-00A REV 001

- **BASELINE SOFTWARE** -00
- **BASELINE VERSION** -01 THUR 99
- **SOFTWARE PROGRAM** -A
- **DOCUMENTATION PACKAGE** D

CPCI VERSIONS

- CPCI VERSIONS ARE NORMALLY CLASS I OR CLASS II CHANGES THAT IDENTIFY VARIATIONS OR MODIFICATIONS OF A CPCl
- CPCI VERSIONS ARE DEVELOPED TO ACCOMMODATE UPDATES TO EQUIPMENT OR BASIC MISSION REQUIREMENTS
- CPCI VERSIONS ARE ALWAYS CO-EXISTENT WITH THE BASELINE CPCl
- A CPIN MUST BE REQUESTED FOR CPCl VERSIONS THAT ARE PRODUCT BASELINED OR DISTRIBUTED TO USING ACTIVITIES
- VERSION CPIN's WILL BE REQUESTED AFTER THE CCB APPROVES THE CLASS I CHANGE

CPIN SUFFIX

NE 81B-ALQ131V/F16A-U0001-00A REV 001

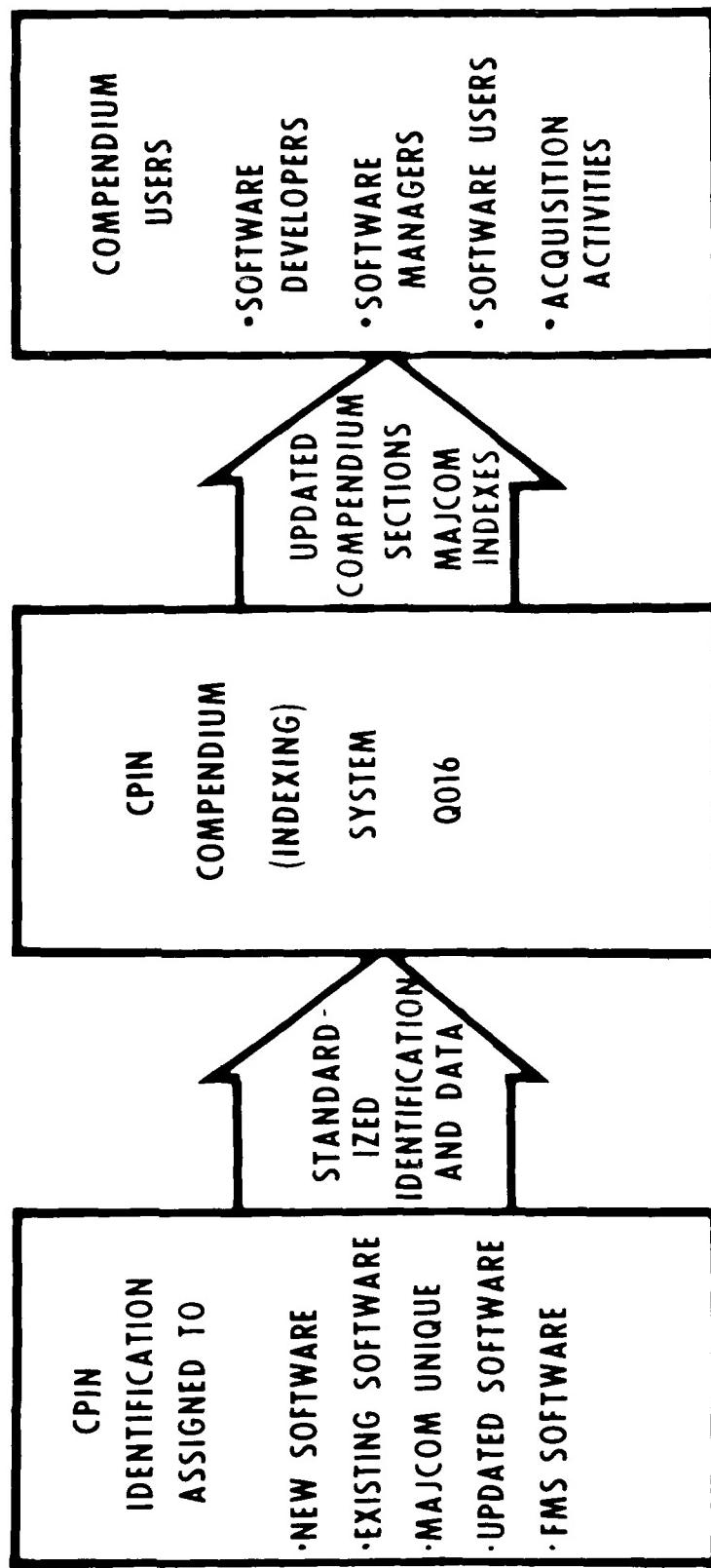
- REVISION NUMBER
- THREE POSITION SERIES NUMBER
- PREFACED WITH "REV"

CPCI REVISIONS

- CPCI REVISIONS IDENTIFY DISCREPANCY CHANGES OR UPDATES IN THE LOGIC OR OPERATION OF A CPCI
- RESULTS OF CLASS I OR CLASS II CHANGE
- CPCI REVISIONS ALWAYS REPLACE THE EXISTING BASELINE CPCI
- CPCI REVISION NUMBERS ARE NOT REQUIRED FOR CPCIs IN ACQUISITION, PRIOR TO PRODUCT BASELINE OR DELIVERY TO USING ACTIVITIES

CPIN COMPENDIUMS

CPIN INDEXING



COMPENDIUM

B ELECTRONIC WARFARE

CATEGORY 81 - AIRCRAFT

ALR69V - AN/ALR-69(V) - COUNTERMEASURES SET, RECEIVING

81B-ALR69V-U001-00A	REV 1, 9 MAY 79	(U)	B	WR
CPU Executer Diagnostic Tape, EQUIP/UUT IDENT 314000-1 Signal Processor CM-479. Tests the central processing unit, A6 CCA, P/N 312059-1 P/O the signal processor (correct operation), 5 Punched Paper Tape, NO. PRGM/CPC 1, LANGUAGE NOVA Assembly, CON COMP/TEST STN 304900-1, -2 SW Proc Test Set AN/APM-379, UUT INTERFACE TEST ADAPTER 314050-1, OPERATOR MANUAL TO 12P3-2ALR69-8-1, OFF-LINE,				
APPL SYS: F-16, F-4D, A-10				
APPL SUBSYS: AN/ALR-69(V), AN/ALR-46.				
TAPE 1 - P/N 314000-1	PRN 1010			
TAPE 2 - P/N 314000-2	PRN 1011			
TAPE 3 - P/N 314000-3	PRN 1020			
TAPE 4 - P/N 314000-4	PRN 1021			
TAPE 5 - P/N 314000-5	PRN 1022			

81B-ALR69V-U001-00D

REV 1, 9 MAY 79 (U)

B WR

DOCUMENTATION PACKAGE CONTAINS: Object and Source ID-310088-1,
Source Listing 104985.

CPCI STATUS INFORMATION

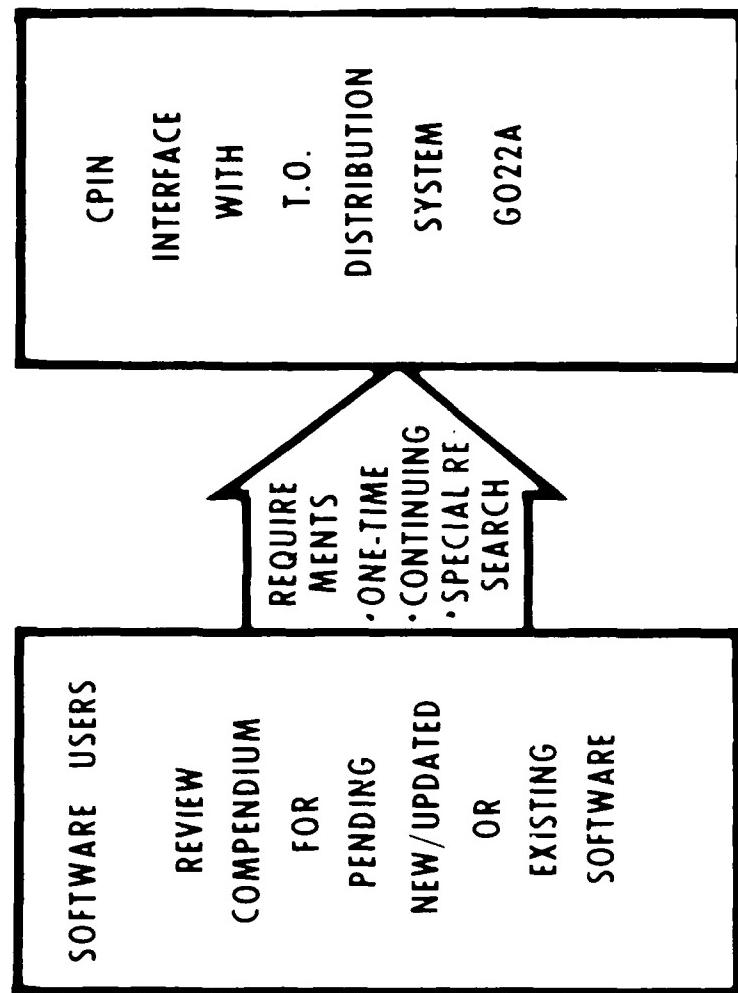
- NEW CPCI (BASELINE)
81B-ALR69V/F16A-U001-00A 15 FEB 79 (U) B WR
- CPCI UPDATE
81B-ALR69V/F16A-U001-00A 15 FEB 79 (U) B WR
REVISION 001 - UNDER DEVELOPMENT
- CPCI REVISION
81B-ALR69V/F16A-U001-00A REV 001, 9 MAY 79 (U) B WR

COMPENDIUMS

<u>MAJOCM</u>	<u>FMS</u>
<u>USAF</u>	
80-1-81	80-3-BE
80-1-82	80-3-DE
80-1-83	80-3-NE
80-1-84	80-3-NO
90-1-85	80-2-SAC
	80-2-SYS
30-1-87	
80-1-88	
80-1-91	

CPIN REQUIREMENTS

CPIN REQUIREMENTS

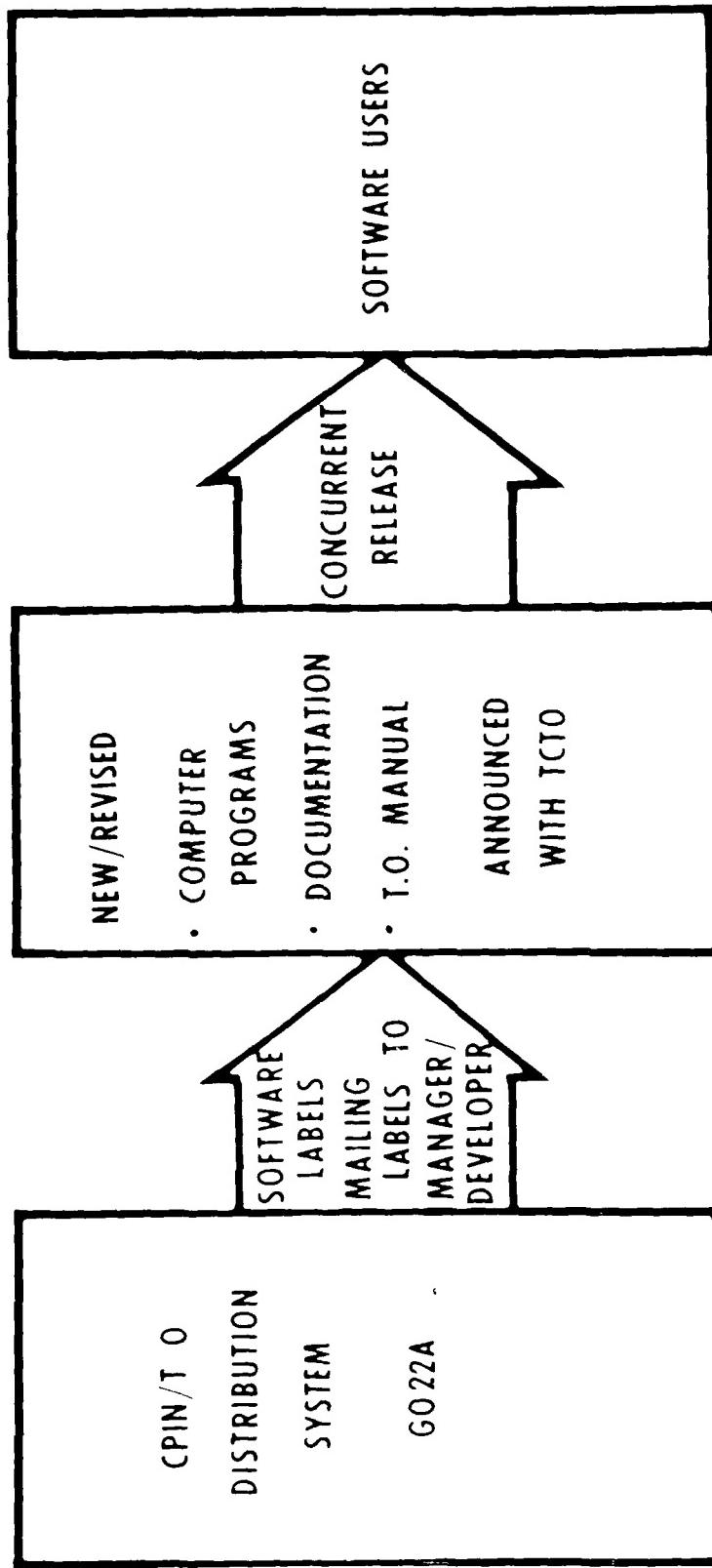


REQUIREMENTS

<u>COMPENDIUM</u>	<u>TODO</u>	<u>RQMTS</u>	<u>ADDRESS</u>
80-1-81	0038	00014	OGDEN ALC/DAPD HILL AFB UT 84056
CPCI	81B-ALR69V/F16A-U001-00A	0066	TAC/ADYC LANGLEY AFB VA 23665

CPII DISTRIBUTION

CPIN DISTRIBUTION



CPIIN DISTRIBUTION

- TWO METHODS:
 - INITIAL DISTRIBUTION
 - ONE TIME REQUISITION
- ACCOMPLISHED BY SOFTWARE MANAGERS
 - ALC/MMEC
 - DOD CONTRACTORS
 - MAJOR COMMANDS
 - OC-ALC/MMEDUC (CPIIN COMPENDIUMS)

LABELS

- ADDRESS LABEL
(2 EACH RQMT)

FROM:	CPIN	QTY
00-ALC/MMEC HILL AFB UT 84056	81B-ALR69V/F16A-U001-00A REVISION 001 - 9 MAY 79	00014
0066	TAC/ADYC LANGLEY AFB VA 23665	

- MEDIA LABELS
(2 EACH RQMT)

CPIN: 81B-ALR69V/F16A-U001-00A, REV 001 DATED: 9 MAY 79
NOUN: RECEIVING COUNTERMEASURE SET P/N: 314000-1
APPLICATION: AN/ALR69(V), F-16A
RELATED MANUAL: I.O. 12P3-2ALR69-8-1
REPLACES: 81B-ALR69V/F16A-U001-00A DATED: 15 FEB 79

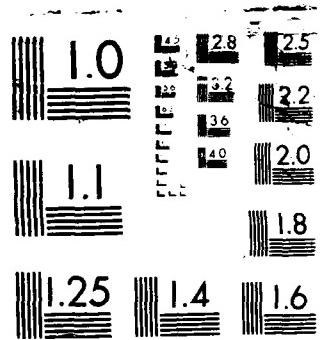
AD-A199 189 JOINT LOGISTICS COMMANDERS' WORKSHOP ON POST-DEPLOYMENT
SOFTWARE SUPPORT (U) JOINT POLICY COORDINATING GROUP 6/6
ON COMPUTER RESOURCE MANAGEMENT JUN 84

UNCLASSIFIED

F/G 12/3

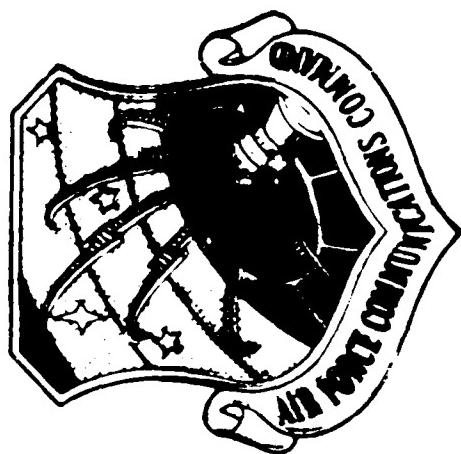
ML

END
DATE
FILED
10 88



CONFIGURATION MANAGEMENT

- TECHNICAL
 - DETERMINE CPCl (USUALLY WBS LEVEL 3 OR 4)
 - DEVELOPMENT/TESTING OF CPCl
 - PROJECT MANAGEMENT FUNCTIONS
 - DISTRIBUTION TO AUTHORIZED USERS
 - CONTROL UPDATES TO CPCl BASELINE
- ADMINISTRATIVE
 - CPIN ASSIGNED TO EACH CPCl/DOCUMENTATION
 - USERS REQUIREMENTS ESTABLISHED/MAINTAINED
 - CPIN ASSIGNED TO EACH CPCl VERSION/REVISION
 - UPDATED CPCls ANNOUNCED WITH TCTO
 - CONCURRENT RELEASE OF CPCl/T0 MANUAL
 - USER REPORTS RECEIPT/IMPLEMENTATION

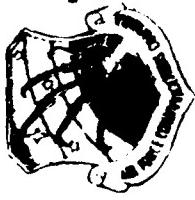


AFDS DC

OVERVIEW

- MISSION
- SCOPE
- ADPS MANAGED
- BROAD FEATURES
- ADS DEVELOPMENT LIFE CYCLE
- REVIEWS, BASELINES, CHANGE CONTROL
- MANAGEMENT CATEGORIES
- SUMMARY





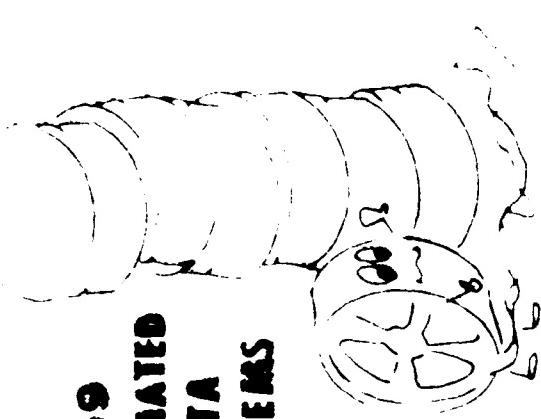
AFSSDC MISSION

PROMOTE ACCOMPLISHMENT OF AF MISSION BY
PROVIDING SPECIFIED AUTOMATED DATA
PROCESSING CAPABILITIES TO MAJOR
COMMANDS AND FIELD UNITS LOCATED AROUND
THE WORLD. THESE CAPABILITIES PERMIT THE
EFFECTIVE AND EFFICIENT ACHIEVEMENT AND
MAINTENANCE OF READINESS, SURVIVABILITY
AND SUSTAINABILITY.

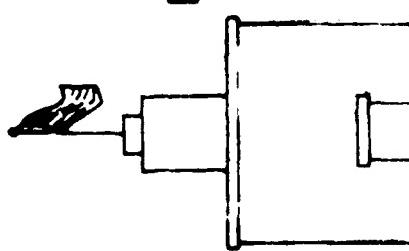


SCOPE

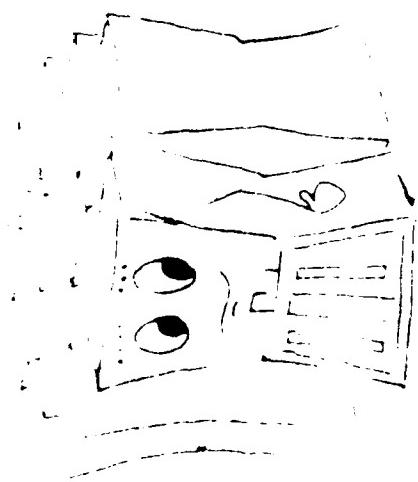
199
AUTOMATED
DATA
SYSTEMS



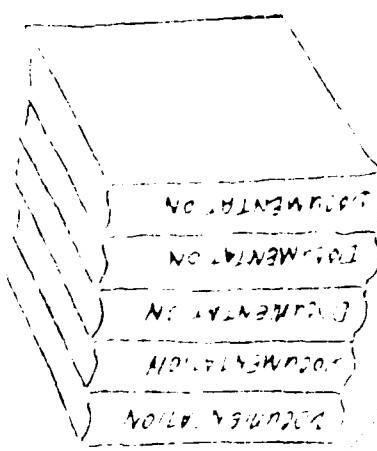
123
BASES



211
SATELLITES



512
COMPUTERS



5,000
PROGRAMS



30,000

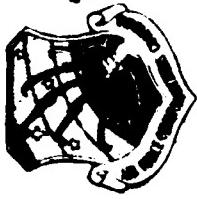
PAGES OF DOCUMENTATION

**AF DATA SYSTEMS DESIGN CENTER
AUTOMATED DATA PROCESSING SYSTEMS
(ADPS)**

<u>ADPS</u>	<u>TITLE</u>
06	STANDARD BASE SUPPLY SYSTEM (U1050-II)
07	BASE LEVEL DATA AUTOMATION STANDARDIZATION PROGRAM (SLOASPI) (S3500-4700)
10	USAF MAJCOM ADP PROGRAM (M6000)
59	BASE LEVEL DATA AUTOMATION PROGRAM (PHASE IV)
80	AIR FORCE WORLDWIDE MILITARY COMMAND AND SYSTEM (WWMCCS) (M6000)

BROAD FEATURES

- PROVIDES POLICY AND PROCEDURES FOR AUTOMATED DATA SYSTEM (ADS) PROJECTS
- STEP-WISE METHODOLOGY
- REVIEWS OCCUR AT KEY DECISION POINTS
- REVIEWS REQUIRED AT ALL LEVELS
- BASELINES ARE ESTABLISHED
- DELEGATES APPROVAL/REVIEW AUTHORITY
- SYSTEM IS FLEXIBLE
- MANAGEMENT ATTENTION WHERE NEEDED MOST
- GEARED TO DELIVERABLE PRODUCTS
 - ALL EFFORTS ARE "VISIBLE"
- CONSOLIDATES AFDS/DC ADS REGULATIONS



(OVERVIEW)

THE DEVELOPMENT LIFE CYCLE

CONCEPTUAL PHASE	DEFINITION PHASE	DEVELOPMENT PHASE	TEST PHASE	RELEASE PHASE	OPERATIONS
DATA AUTHORITY, DOCUMENTED, JUSTIFIED AND APPROVED.	COMPLETE DEFINITION OF SYSTEM DESIGN AND SYSTEM DETAILS.	COMPLETE DESIGN OF SUBSYSTEMS, CODED AND ESTABLISHED.	TEST THE ENTIRE SYSTEM LOCALLY AND IN THE FIELD.	RELEASE AND USE IN THE FIELD.	PRODUCT BASELINE.
DATA AUTHORITY, DOCUMENTED, JUSTIFIED AND APPROVED.	DETAILED DESIGN AND SYSTEM DESIGN.	COMPLETE DESIGN OF SUBSYSTEMS, CODED AND TEST PROGRAMS.	TEST THE ENTIRE SYSTEM LOCALLY AND IN THE FIELD.	RELEASE AND USE IN THE FIELD.	PRODUCT BASELINE.
DATA AUTHORITY, DOCUMENTED, JUSTIFIED AND APPROVED.	DETAILED DESIGN AND SYSTEM DESIGN.	COMPLETE DESIGN OF SUBSYSTEMS, CODED AND TEST PROGRAMS.	TEST THE ENTIRE SYSTEM LOCALLY AND IN THE FIELD.	RELEASE AND USE IN THE FIELD.	PRODUCT BASELINE.
DATA AUTHORITY, DOCUMENTED, JUSTIFIED AND APPROVED.	DETAILED DESIGN AND SYSTEM DESIGN.	COMPLETE DESIGN OF SUBSYSTEMS, CODED AND TEST PROGRAMS.	TEST THE ENTIRE SYSTEM LOCALLY AND IN THE FIELD.	RELEASE AND USE IN THE FIELD.	PRODUCT BASELINE.

REVIEWS, BASELINES, CHANGE CONTROL

CONCEPTUAL PHASE	DEFINITION PHASE	DEVELOPMENT PHASE	TEST PHASE	OPERATIONS PHASE
PRELIMINARY REQUIREMENTS REVIEW (PRR) SYSTEM/SUBSYSTEM REQUIREMENTS REVIEW (SSR) FUNCTIONAL BASELINE	SYSTEM DESIGN REVIEW (SDR) ALLOCATED BASELINE	PRELIMINARY DESIGN REVIEW (PDR) CRITICAL DESIGN REVIEW (CDR) PRELIMINARY FUNCTIONAL CONFIGURATION AUDIT (FCA)	EST-1 EST-11 FINAL FCA FINAL PCA SYSTEM VALIDATION REVIEW (SVR)	FINAL OPERATIONAL EVALUATION (FOE)

→ SPECIAL MANAGEMENT REVIEW (SMR) →

MANAGEMENT CATEGORY I

- PROJECTS
- APPROVAL AUTHORITY
- REVIEW AUTHORITY
- DIVISION CHIEF
- BRANCH CHIEF
- PROGRAM MAINTENANCE
OR SYSTEM MODIFICATION
3 MARYEARS OR LESS

MANAGEMENT CATEGORY II

APPROVAL REVIEW
AUTHORITY

DIVISION CHIEF

DIRECTOR

PROJECTS

- ADS MAINTENANCE OR SYSTEM MODIFICATION GREATER THAN 3 MANYEARS BUT LESS THAN OR EQUAL TO 10 MANYEARS

- DEVELOPMENT OF NEW COMPUTER PROGRAMS / SUBSYSTEMS FOR EXISTING ADS

- NEW ADS DEVELOPMENT UP TO 10 MANYEARS

MANAGEMENT CATEGORY III

PROJECTS	APPROVAL AUTHORITY	REVIEW AUTHORITY
• ALL PROJECTS EXCEEDING 10 MANYEARS OR:	TECH ASSISTANCE TO THE COMMANDER	TECH ASSISTANCE TO THE COMMANDER
– REQUIRE ACQUISITION OF ADPE AT AFPSDC		
– REQUIRE ADDITIONAL DIRECTORATE MANPOWER		
– REQUIRE CONTRACTUAL SERVICES		
– REQUIRE COMMERCIAL SOFTWARE ACQUISITION		
– REQUIRE MAJCOM / SOAs TO ACQUIRE ADPE		
– REQUIRE BASIC SOFTWARE DEVELOPMENT / MODIFICATION		
– SPECIAL INTEREST		
– DEVELOPED AS JOINT DIRECTORATE OR MAJCOM / SOA EFFORTS		

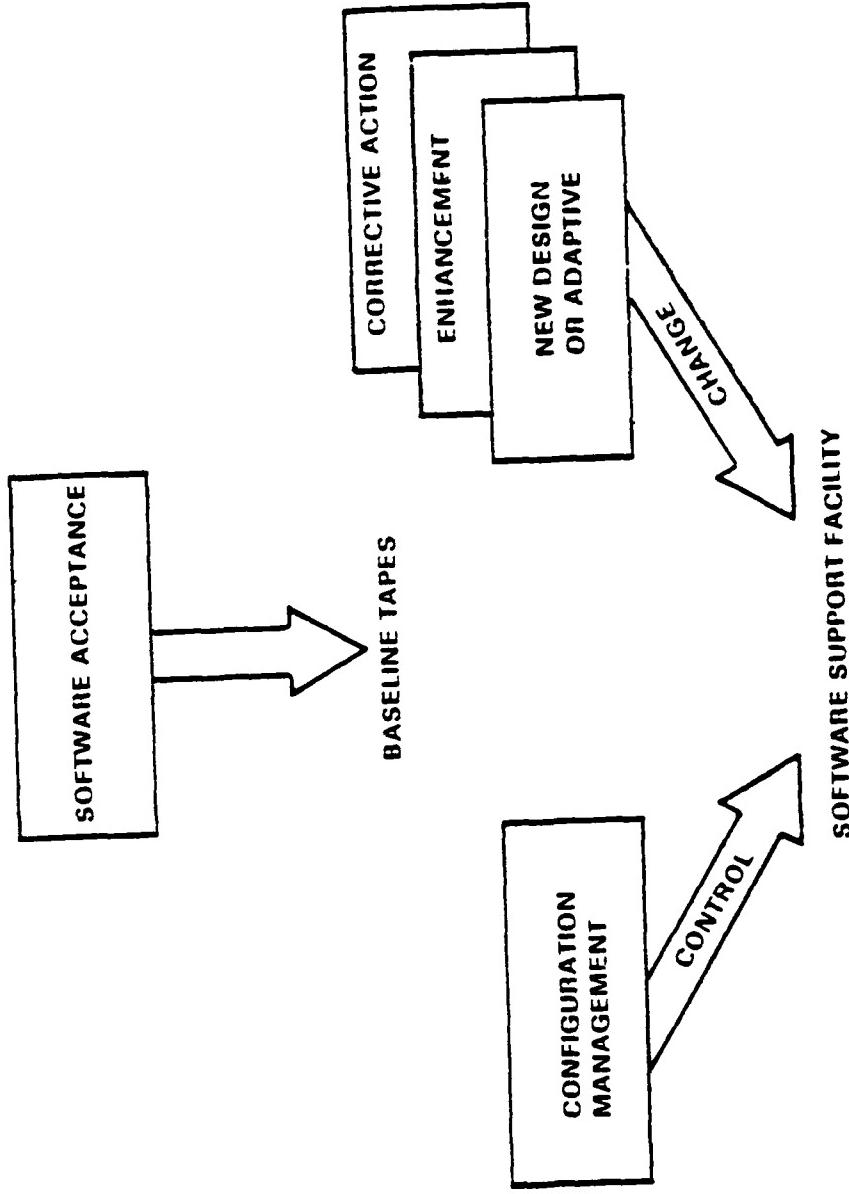
BRIEFING SUMMATION

- FORMALIZES/STANDARDIZES THE METHODOLOGY.
- BREAKS THE DEVELOPMENT CYCLE INTO VISIBLE, UNDER-STANDABLE MANAGEABLE PORTIONS (PHASES, STEPS).
- ASSUMES ADEQUATE REVIEW AT CRITICAL POINTS IN THE DEVELOPMENT CYCLE. (DEFINE BEFORE DESIGN.)
- SYSTEM IS FLEXIBLE.
- RESPONSIVE TO THE AIR FORCE USER.
 - REQUIREMENTS DEFINITION
 - USER PARTICIPATION IN DEVELOPMENT CYCLE
 - TIMELINESS
 - RELIABILITY

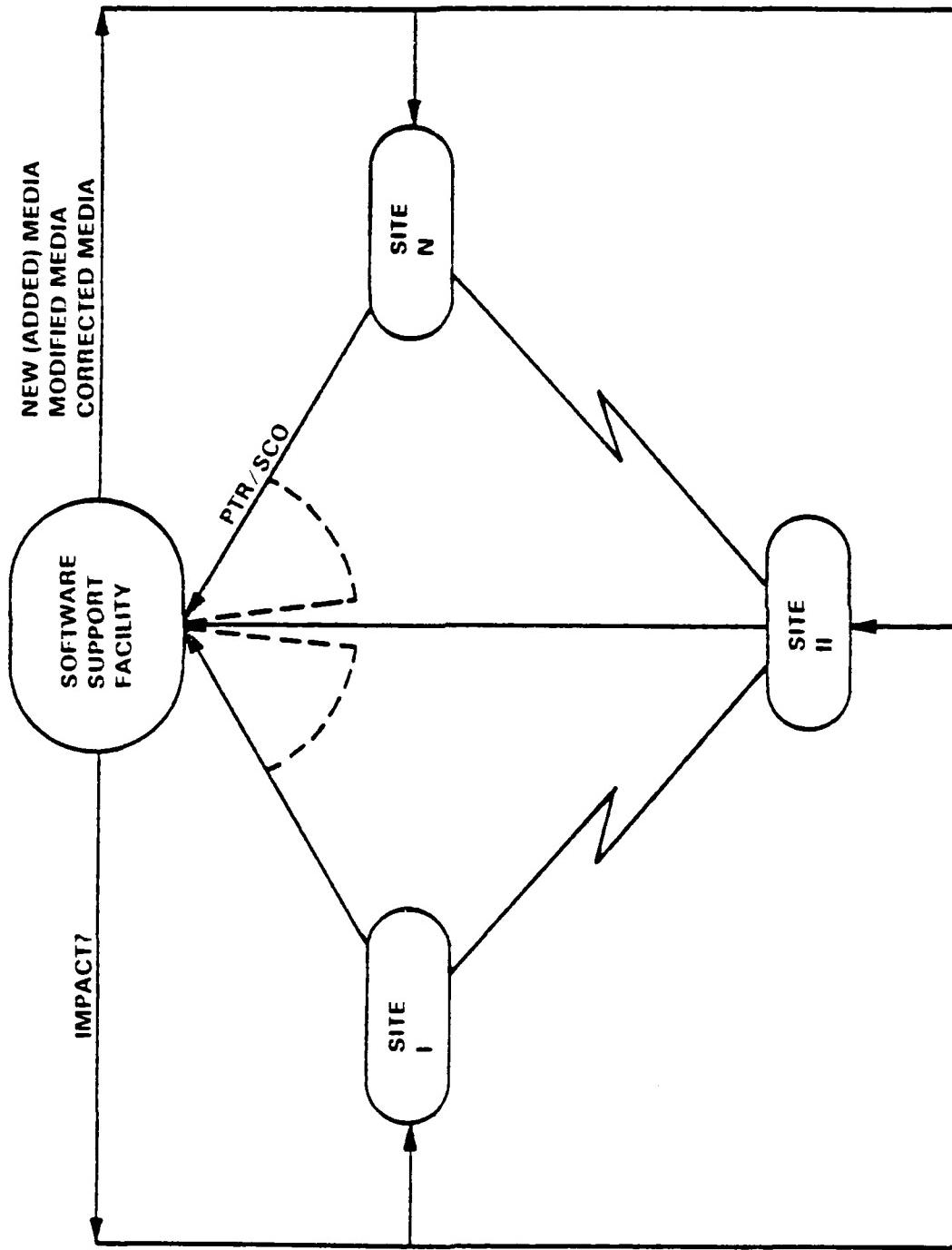
VI

- OPERATIONAL AND MAINTENANCE
CONFIGURATION MANAGEMENT

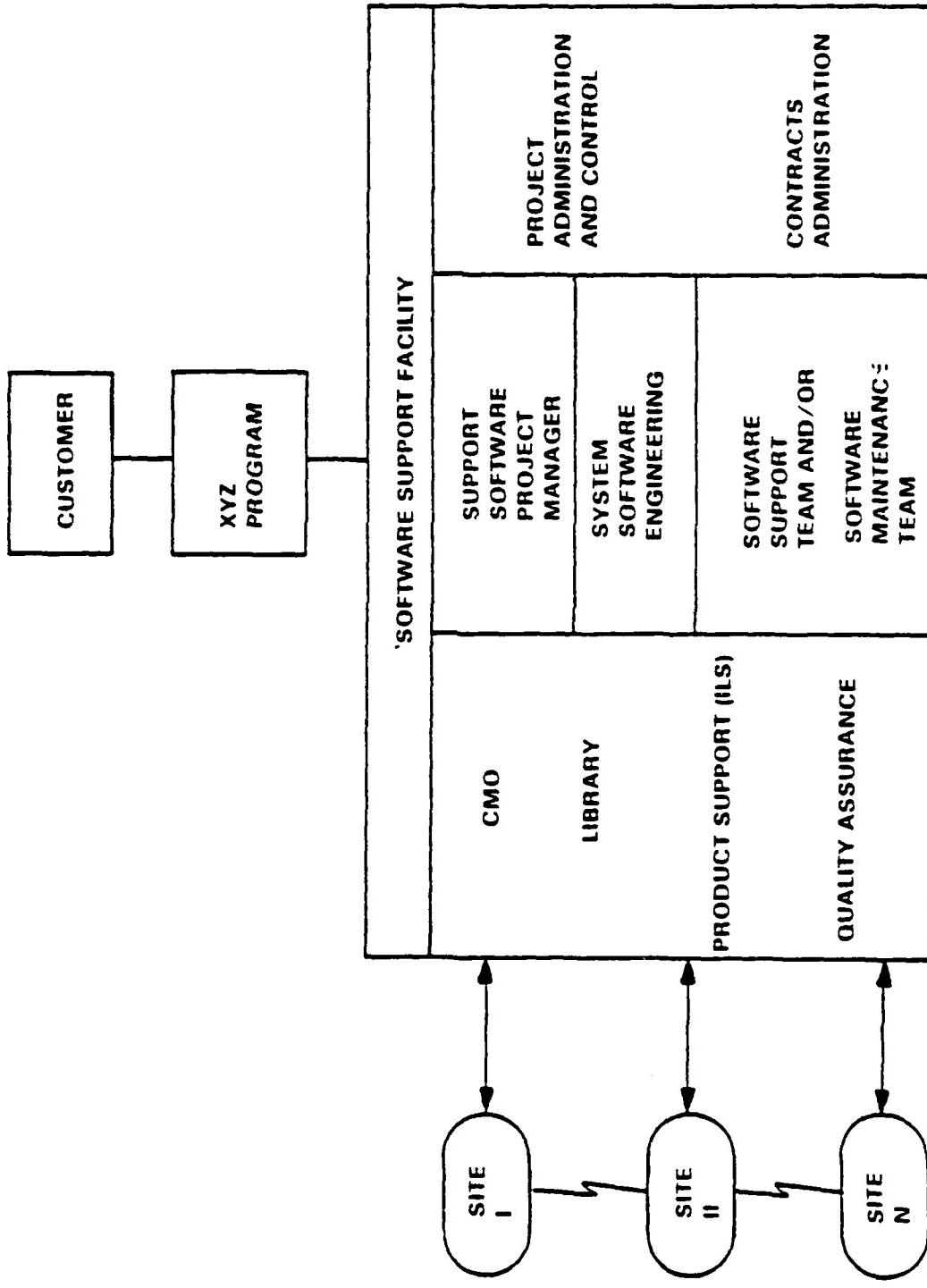
OBJECTIVES



FUNCTIONAL FLOW COMMUNICATION AND CHANGES



FUNCTIONAL ORGANIZATION



RULE: NO CHANGES MADE IN THE FIELD

O&M AUDITS

- DOCUMENTS:
 - ▲ PERFORMANCE/DESIGN SPECIFICATIONS
 - ▲ DATA REQUIREMENTS
 - ▲ INTERFACE DESIGN SPECIFICATION
 - ▲ DATA BASE SPECIFICATION
 - ▲ TEST AND IMPLEMENTATION PLAN
 - ▲ USER MANUAL
 - ▲ COMPUTER OPERATION MANUAL
 - ▲ TEST ANALYSIS REPORTS
 - ▲ TEST DATA
 - ▲ COMMERCIAL COMPUTER MANUALS
- ▲ SOFTWARE LEVEL BREAKDOWN

O&M AUDIT

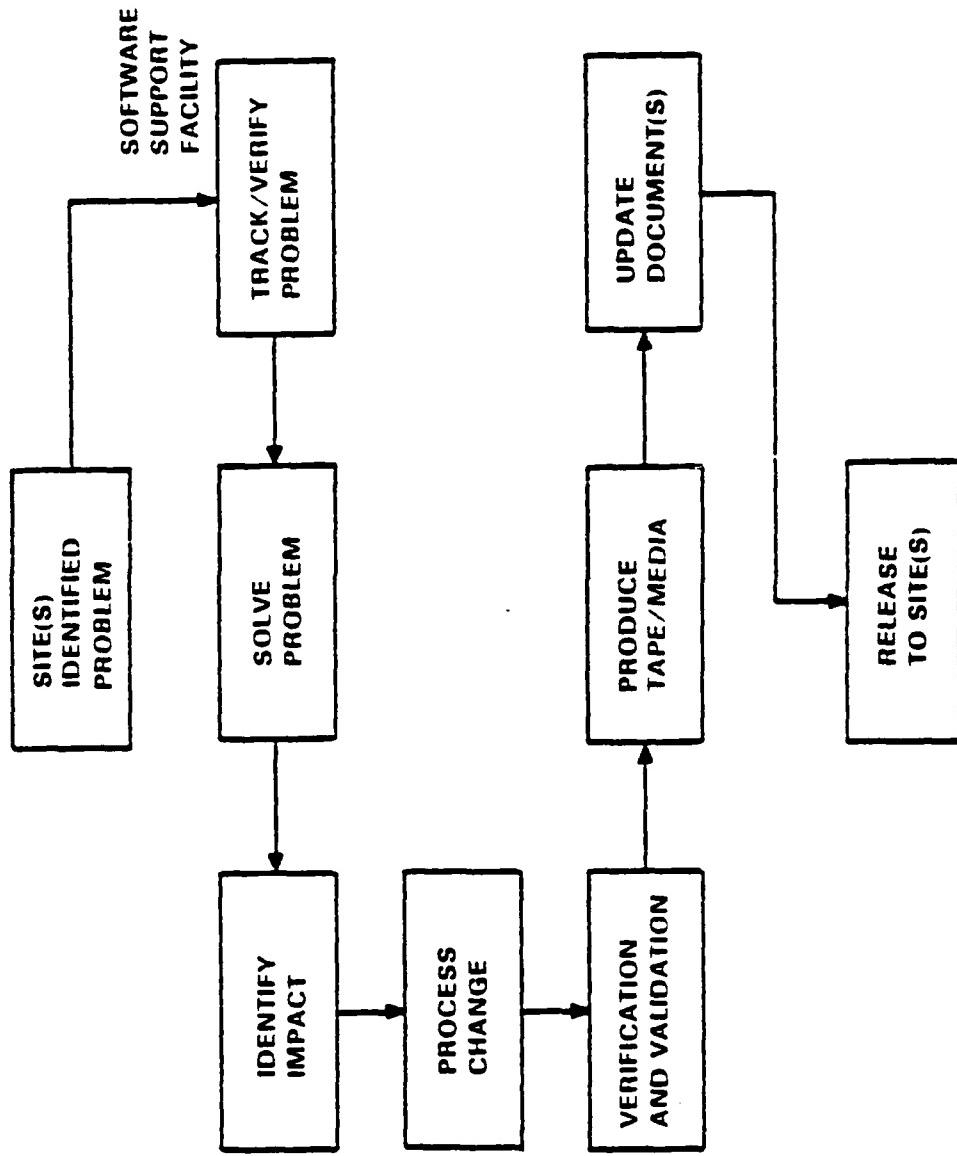
- SOFTWARE
 - ▲ OBJECTIVES
 - DOCUMENTS TRACE TO SOFTWARE PROGRAMS
 - SOFTWARE MASTER EQUALS THE CURRENT REVISION
 - PROGRAMS PERFORM TO RELATED DOCUMENT
 - TEST DATA SHEETS VERIFY PERFORMANCE
 - RECONCILE DISCREPANCIES

SA A SANDERS
ASSOCIATED INC.

CONTROLS

- TYPES OF CHANGE ACTIVITY
 - ▲ CORRECTIVE ACTION
 - FAILURES
 - TROUBLE
 - ▲ ADAPTIVE
 - MODIFICATIONS
 - PLANNED ADDITIONS
 - ▲ ENHANCEMENTS
 - IMPROVEMENTS

CHANGE PROCESS





COMPUTER SOFTWARE CHANGE PROPOSAL			
DOCUMENT/PROGRAM NO 211M6 001 CP 140	REV BITE PROGRAM	DESCRIPTION/WHICH BITE PROGRAM	COP IDENT. NO 94117
		DATE 8/15/78	
1. DESCRIBE CHANGE AND SENDING INITIALIZATION COMPLETE IF INTERRUPT TO SS0 47 PROGRAM CALL 2. REASON FOR CHANGE AIP SYSTEM DOES NOT MEET INTERFACE SPECIFICATION UOFS NOT NOW SEND INTERRUPT WHICH CAUSES OPDC TO HANG			
3. PROGRAMS AFFECTED SS0 47 6924550 Q1 REV A PROGRAM 5924550 P1 REV A OBJECT FILE 6924551 P1 REV A SOURCE FILE 6924552 P1 REV A BINARY FILE			
4. SOFTWARE DOCUMENTS AFFECTED FUNCTIONAL/PERFORMANCE SPEC DESIGN SPEC DATA BASE/INTERFACE SPEC MANUALS BASELINE			
5. RECOMMENDED APPROVALS RECOMMENDED PRELIMINARY AIP			
6. RELATIONSHIP TO OTHER CHANGES NONE			
7. POST ESTIMATE OF CHANGE \$ 660 N/R 0 RECURRING SCHEDULED NONE			
8. IMPACT 1) BENCHMARK 2) CUSTOMER SPEC 3) SIMCONTRACT 4) HARDWARE			
9) EQUIP SWITC 0) EQUIP HARDWARE 1) EQUIP/HARDWARE COMPUTER 0) FACILITY/SITE			
CHANGE LEVEL TO 0) CHANGE P/Q TO 1) CHANGE PART NO 0) OTHER			
B 02/P2 N/A			
PREPARED BY J. SMITH	DATE 8/15/78	COMMITMENT APPROVAL	DATE 8/15/78
CONFIRMED SA NO 100-000-000		CONFIRMED ORGANIZATION: WHITING/PACIFIC VELVET/WH CONINCO/PINK REQUESTER	





SOFTWARE TROUBLE REPORT

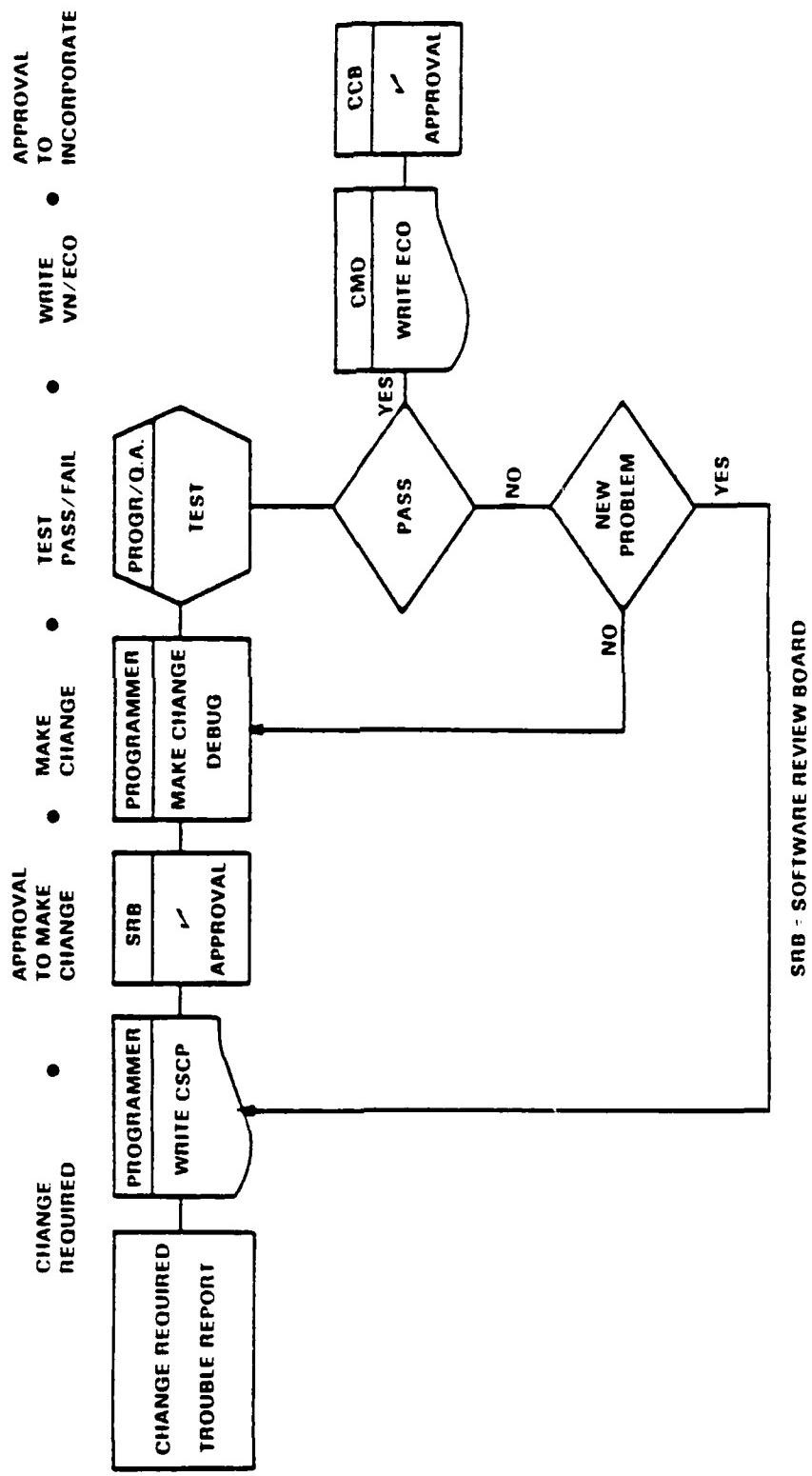
SIR

TITLE/DESCRIPTION		CONTRACT	REPORT DATE
PROBLEM DATE/TIME	8/18	COMPUTER/terminal	OPERATING SYSTEM
TEST NAME	FAILED IN SPECIFIED WINDOWS SPEC	VERSION	
PROBLEM DESCRIPTION			

ACTION REQUEST BY DIVISION NO. 0

SUBMITTED BY DATE	ASSIGNED TO DATE	APPROVED BY DATE
RAIS MC DICHO	BY	PRIORITY
DISPOSITION		
CLOSED BY		DATE
8/20 1989 11:10		

SOFTWARE CHANGE METHODOLOGY



SNB - SOFTWARE REVIEW BOARD

CONFIGURATION CONTROL BOARD (CCB)

- ESTABLISHED AT SSF
- COMPLETE AUTHORITY
- RESPONSIVE TO EMERGENCY AND URGENT CHANGES

CCB MEMBERS

- SOFTWARE SYSTEMS CONFIGURATION MANAGER--CHAIR
 - SOFTWARE SYSTEMS LIBRARIAN--CLERK
 - SOFTWARE SYSTEMS ENGINEER
 - SOFTWARE SYSTEMS QUALITY ASSURANCE
 - SOFTWARE SYSTEMS CHIEF PROGRAMMER
 - HARDWARE SYSTEMS ENGINEER
 - SYSTEMS OPERATIONS OFFICER
 - CUSTOMER REPRESENTATIVE
- EMERGENCY SQUAD

STATUS ACCOUNTING

- UPDATE BASELINE TO LATEST REVISION
- HISTORY OF CHANGES BY DATE, NUMBER
- TRACEABILITY OF CHANGES

IF YOU MUST HAVE CONTROL AT SITES

- CHIEF OR LEAD PROGRAMMER
 - ▲ REVIEW AND CONCURRENCE
- LIBRARIAN FUNCTION - WITH NECESSARY DOCUMENTATION AND RECORDS.

APPENDIX D
ORLANDO I
CONFIGURATION MANAGEMENT PANEL CHARTER

1. ISSUE

Each of the three services approaches PDSS Configuration Management (CM) from slightly different perspectives. Possibly this has resulted from an attempt to apply reasonably mature hardware CM practices to software. The elements of CM: configuration control, identification and status accounting are all as equally important in the software world as in the hardware world. Configuration control is even more important when the software is firmware in a Read Only Memory device. Changes to firmware installed in a piece of equipment could change the hardware configuration of the equipment.

The major issue which the panel will address is determining the scope of Software Configuration Management. A multi-service definition to be promulgated by the JLC will be one of the products of our efforts.

Specific issues which this panel will address are:

1.1 Issue A. Baseline Definition - Different services treat the transition from development to operational status differently. The question of when a system becomes operational needs to be resolved. The nonstandardization of requirements and procedures for transition to PDSS agency needs to be resolved. When multiple services are involved in the same program or when a single service delivers production items to two or more services for use, baseline definition and control is confusing.

In the operational phase of the life cycle, multiple baselines may exist. Configuration management is critical at this time.

1.2 Issue B. Configuration Management Scope and Terminology -

Automated military systems frequently have multiple computers, systems, displays, weapon mix and software components. Configuration management is done individually on hardware components of the system. Software changes can cause the system capability to change which alters the configuration but may not change the nomenclature or identification of the entire system.

Managing the configuration of software when various versions of a system are deployed is a problem. Identification of each version is important. The evolving development of deployed automated systems creates hardware diversity. Software changes compound this diversity. The relationship of the development Program Management Office to the PDSS agency under such conditions needs to be clarified.

Configuration management of interfaces among systems, especially when they are in different phases of the life cycle or are maintained by different agencies, services or nations, is a perplexing and ill-defined issue.

1.3 Issue C. Software Security - Different projects, even within individual services, treat the classification of object code, source code, storage media (including RAM and ROM) in different ways. Facility security is also treated differently between projects. Software security is not understood by security-type people. Clarification, guidance is required.

1.4 Issue D. Methods of Tracking the Configuration and Need for a Central/Backup Repository for Software. Generally, the only repository for software (tapes, documentation, source/object code) resides with the Software Support Activity (SSA). Consideration should be given to the hazards of this single site storage.

Automated methods of tracking the configuration and the physical control of this data base needs to be addressed.

Configuration management of technical data (manuals, drawings, etc.) must be addressed, especially as it relates to releases of the software.

2. QUESTIONS

The purpose of these questions/discussion points is to stimulate discussion of as many facets of the issues as possible. They are certainly not all inclusive and others will be posed by the panel as they explore their areas. Some may not be germane.

2.1 Issue A. Baseline Definition

1. Purpose of baselines - who uses them?
2. What do they contribute to PDSS process?
3. What is the operational baseline?
4. What should be turned over to the SSA from the developer?
5. What should be the situation for declaring that a baseline is operational?
6. How can CM deal with multiple baselines during the operational phase?
7. When should the SSA take over baseline management?

2.2 Issue B. Configuration Management Scope and Terminology

1. What role should the PDSS activity perform in order to support the configuration management of the military system as a whole?
2. How should the nomenclature of systems be defined to reflect differences in software?
3. How can the PDSS agency influence changes caused by development of new systems that affect the CM of a deployed system under control of the agency?

4. How is configuration management to be performed on interfaces among systems?
5. To what level must the decision on interface changes be raised?
6. How can division of responsibility among PDSS agencies for support of different components of a system be defined and maintained?
7. What can the PDSS agency do to influence the development process to make PDSS easier?
8. Who defines the software CI and when?
9. What comprises a software CI?
10. Is it desirable for all DoD to settle on a standard/common CPIN?
11. Should/can the CPIN fit into the hardware automated identification systems?
12. What is the content (not format) of CPIN record?

2.3 Issue C. Software Security

1. How should source code be classified?
2. How should object code be classified?
3. How about equipments which contain classified software in ROM, RAM, etc.?
4. Physical security requirements for host systems?
5. How is classification to be determined and marked? (System, chips, tapes, etc.)
6. How is security to be verified?
7. Facility security requirements?

2.4 Issue D. Tracking and Storage Methods

1. How will software CM tracking be done?
2. Is an automated data base necessary?
3. Where should this data base be located? Redundancy?
4. Is software CM tracking different from hardware or could the same methods be used for both?
5. Is it desirable to transition the developer's CM methods to the PDSS agency?
6. Should all PDSS agencies use the same method?
7. Does this require constraints on the development contract?
8. What gets transferred from the developer to the SSA?
9. What backup is necessary to the primary software repository and under what conditions?
10. Where should it be: DoD activity, contractor (original or other) or both?
11. What is to be maintained under configuration control (technical data, software, other)?
12. How is configuration control to be implemented?

3. WORKSHOP METHODOLOGY

3.1 First Day

After the opening Workshop General Session on Monday afternoon, the six Workshop panels will meet in their assigned rooms for the first time.

NOTE: For purposes of this outline, meetings of all Workshop participants will be referred to as "General Sessions" while meetings of all members of our panel will be referred to as "Group Sessions."

The purpose of the initial Group Session will be:

- To review the Panel's objectives and purpose;
- to stress the requirement to provide objective, well-defined recommendations to the JLC/CRM on the issue of Software Configuration Management;
- to discuss general approach to the Panel's operation, schedule, administration detail, room locations, etc.);
- to discuss the planned approach to use subpanels and the group sessions;
- to allow the Co-Chairs and each member of the Panel to introduce herself/himself to the group.

Further, it is planned that selected members of the Panel, or invited guests (if appropriate) will be contacted prior to the Workshop and requested to present briefings on selected topics. Briefings would be from 15-20 minutes in length and would provide:

- NTEC presentation on Trainer CM;
- Descriptions of how PDSS CM is presently done by various services and selected industries;
- Current governing MIL-Standards, regulations and directives;
- Known problems and proposed solutions.

Suggestions for other briefings will be welcome and should be provided on the survey form to be returned.

3.2 General Panel Operation

Beginning Tuesday morning, we will divide into four subgroups, each with a leader, to address the following issues:

Subgroup A - Baseline Definition

Subgroup B - Configuration Management Scope
and Terminology

Subgroup C - Software Security

Subgroup D - Tracking and Storage Methods

Wherever possible members will be allowed to select the sub-panel/topic area of their choice. Subpanel leaders and recorders will be selected by the subpanel members in cooperation with the panel co-chairs.

Subpanels will discuss their assigned topic areas, identify planned recommendations in accordance with this outline, and prepare written notes on major items of discussion/decisions.

Issues of a general nature, or those which may have the potential for impacting another subpanel and/or panel, will be identified by the subpanel leader and reported to one, or both, panel co-chairs. The co-chairs will facilitate/coordinate required interaction.

Subpanels will meet all day on Tuesday, Wednesday, and Thursday.

3.3 Daily Afternoon Workshop Group Sessions

The subpanels will reform for a short group session late in the afternoon on each day (Tuesday, Wednesday, Thursday). Subpanel leaders and/or recorders will provide a brief (3-5 minute) review of the panels deliberations, and report on the preliminary recommendations developed by the panel.

The co-chairs will prepare, based on these subpanel reports, a panel summary for presentation at the following General Session.

3.4 Daily Afternoon General Session

The Workshop body will meet in General Session each afternoon on Tuesday, Wednesday, Thursday, and for a final session on Friday morning. At these sessions, each Workshop panel will provide a brief report on their deliberations and preliminary recommendations. These presentations will be made by the panel co-chairs.

The General Sessions are designed to provide the Workshop Committee, the Co-Chairs and all Workshop participants with an assessment of progress, to surface requirements for inter-panel coordination, to provide information required to redirect or reinforce panel focus (if required), and to provide all Workshop participants with a view into the total Workshop activities.

3.5 Final Workshop General Session - Friday

For the final Workshop Session, each of the panels will prepare and present a 15-20 minute briefing. This briefing will summarize the panel's work, our recommendations, as well as providing other salient information.

This briefing and the summary written report will be prepared by the panel co-chairs and subpanel chairs Thursday night.

3.6 Written Report

The subpanel chairs will prepare reports on their efforts and the panel co-chairs will coordinate these and write the preliminary report of the panel's activity. This will be distributed to all panel members for their comments (including minority reports) before being submitted to the JLC.

Based on comments from the JLC, the panel chairs will prepare a final Configuration Management Report which will present details on the Panel's Charter, deliberations and recommenda-

tions. The full Workshop report will be distributed to all participants.

4. SUGGESTED PRODUCTS

One major product of our panel will be a definition of the scope of PDSS Configuration Management.

4.1 Subpanel A. Baseline Definition

1. Definition of operational baseline;
2. Definition of when the software becomes operational;
3. Recommended CM transition plan from developer to SSA;
4. Recommendation for dealing with multiple, parallel baselines;
5. Other.

4.2 Subpanel B. Configuration Management Scope and Terminology

1. Definition of PDSS configuration management scope;
2. Policy recommendation statement of inter- and intra-PDSS configuration management relationships;
3. Recommended CPIN system and identification of implementing standard;
4. Recommended policy statement on extend of PDSS configuration management influence in the development process.
5. Recommended guidelines on S/W configuration item definition.
6. Other.

4.3 Subpanel C. Security

1. Policy recommendation concerning software security processes, classification determination, verification, handling procedures, etc., for inclusion in DoD/individual services security manuals, directives, etc.;
2. Other.

4.4 Tracking and Storage Methods

1. Definition of the physical CM products and what must be maintained under configuration control;
2. Policy recommendation on the need for repository for backup storage of software and pertinent documentation;
3. Identification of what must be transferred from developer CM efforts to the PDSS agency;
4. Groundrules for an Automated Tracking and Status Accounting System (when and under what conditions would it be necessary);
5. Other.

MANAGEMENT CONTROL

(RESOURCE CONTROL
FUNCTION)

ASSI
CONT
ACCE
PRI

=====
=====
DOCUMENTATION

CONFIGURATION

MANAGEMENT

COLLATE
BAK
DOCUM
AND 1
CHANGE
DOCUM

=====
=====
**ENGINEERING
FUNCTION**

CHANGE

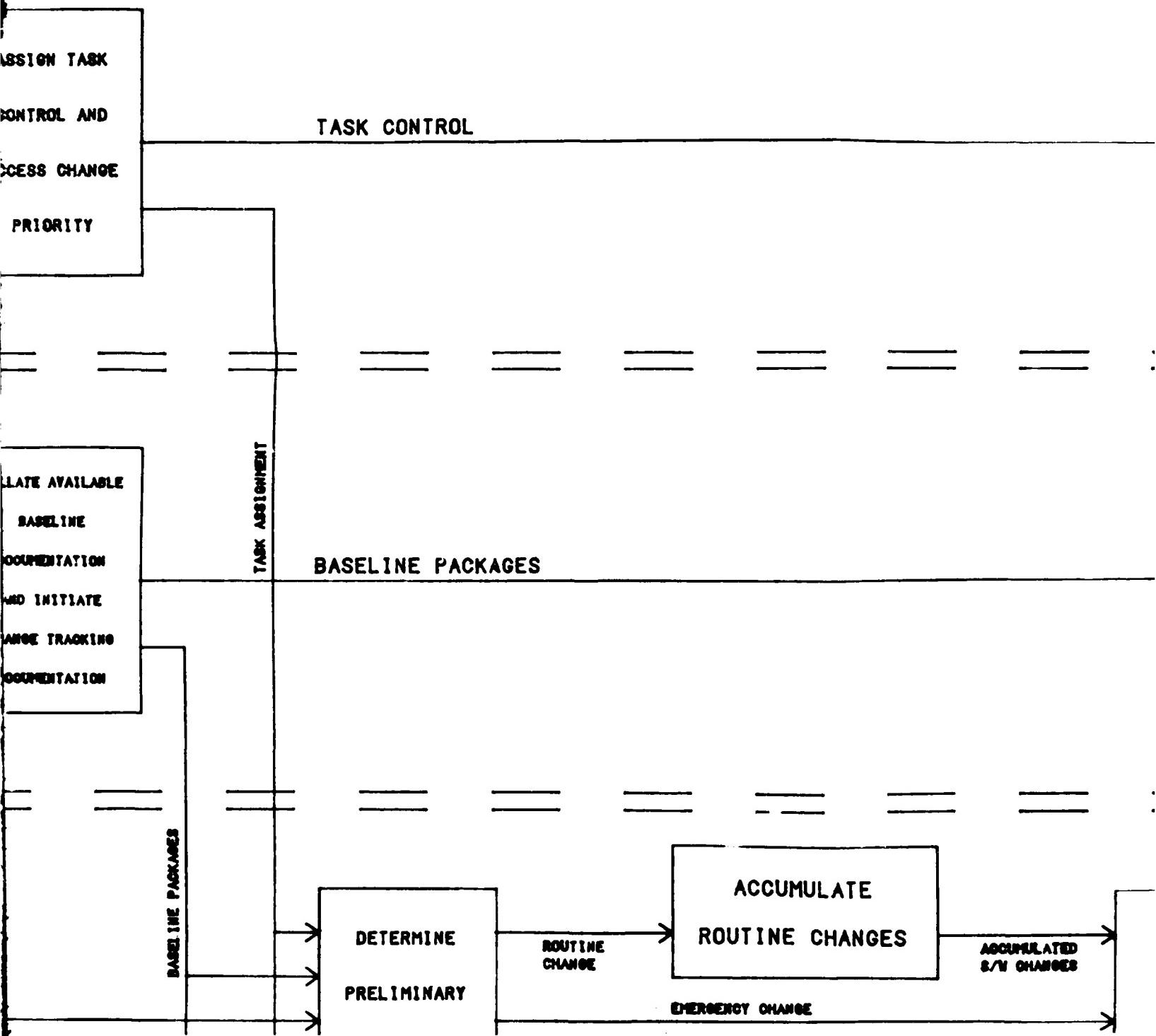
→

ORIGINATOR

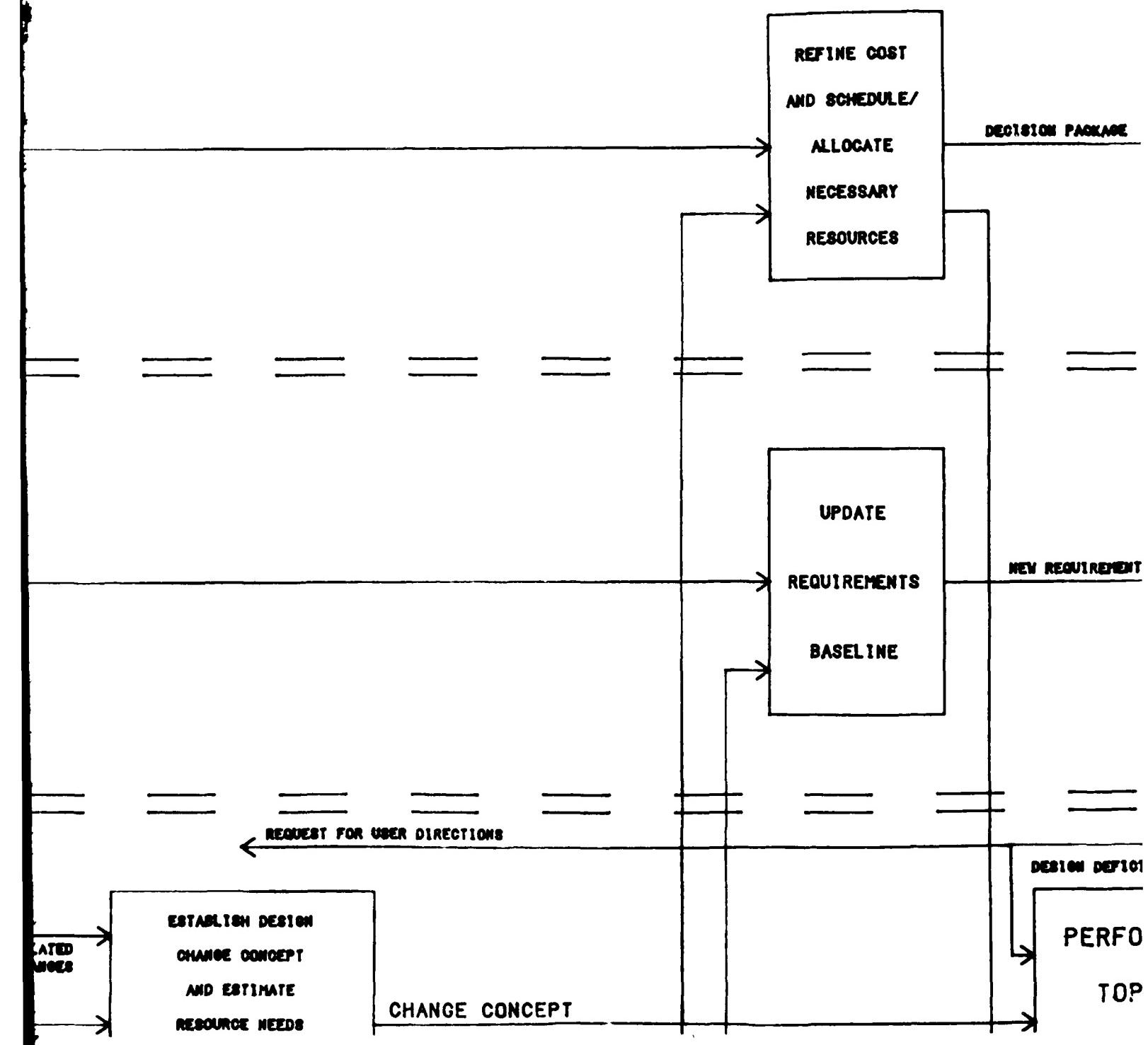
RECEIPT OF

ORIGINATOR

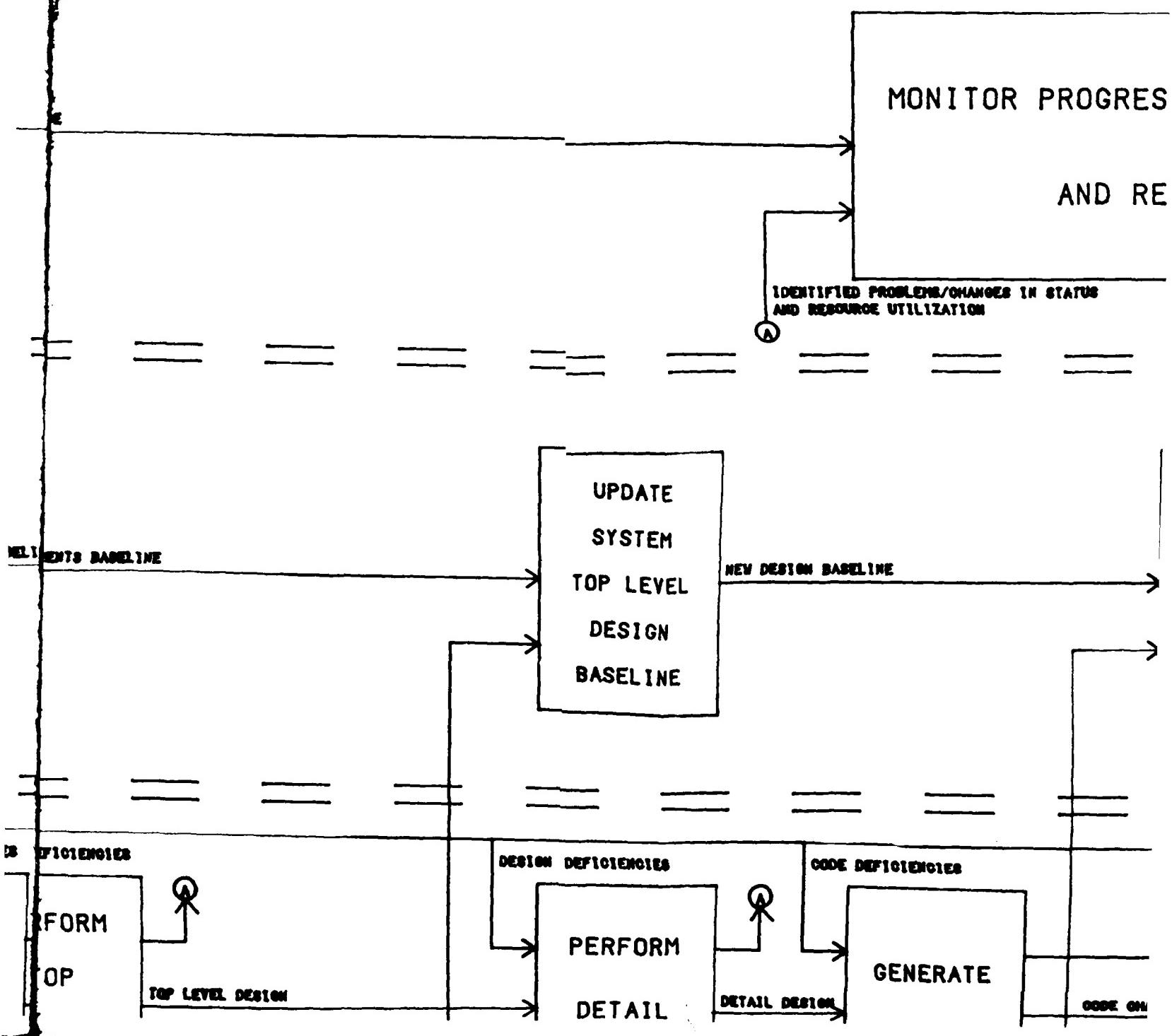
GENERATION



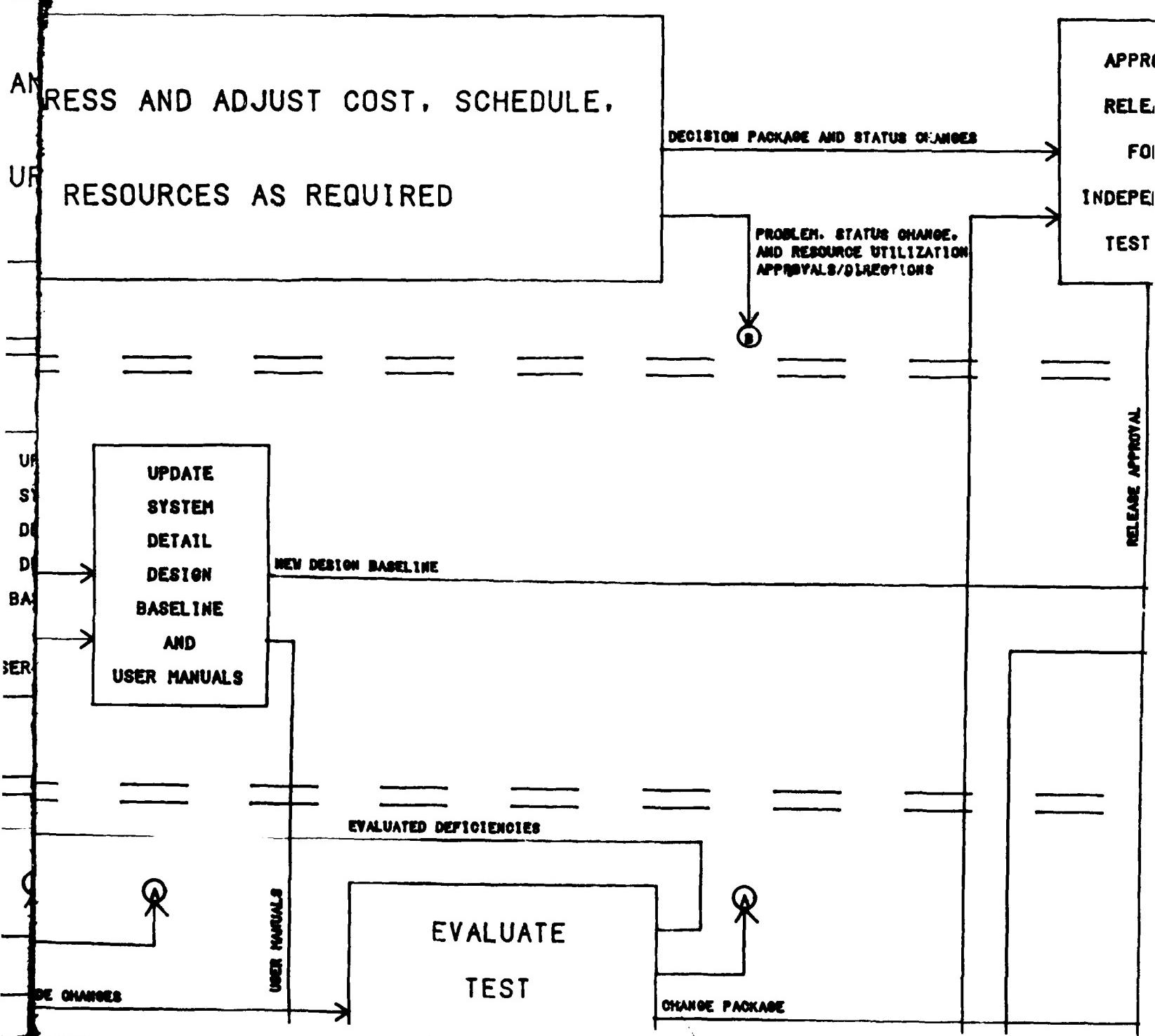
IC SOFTWARE



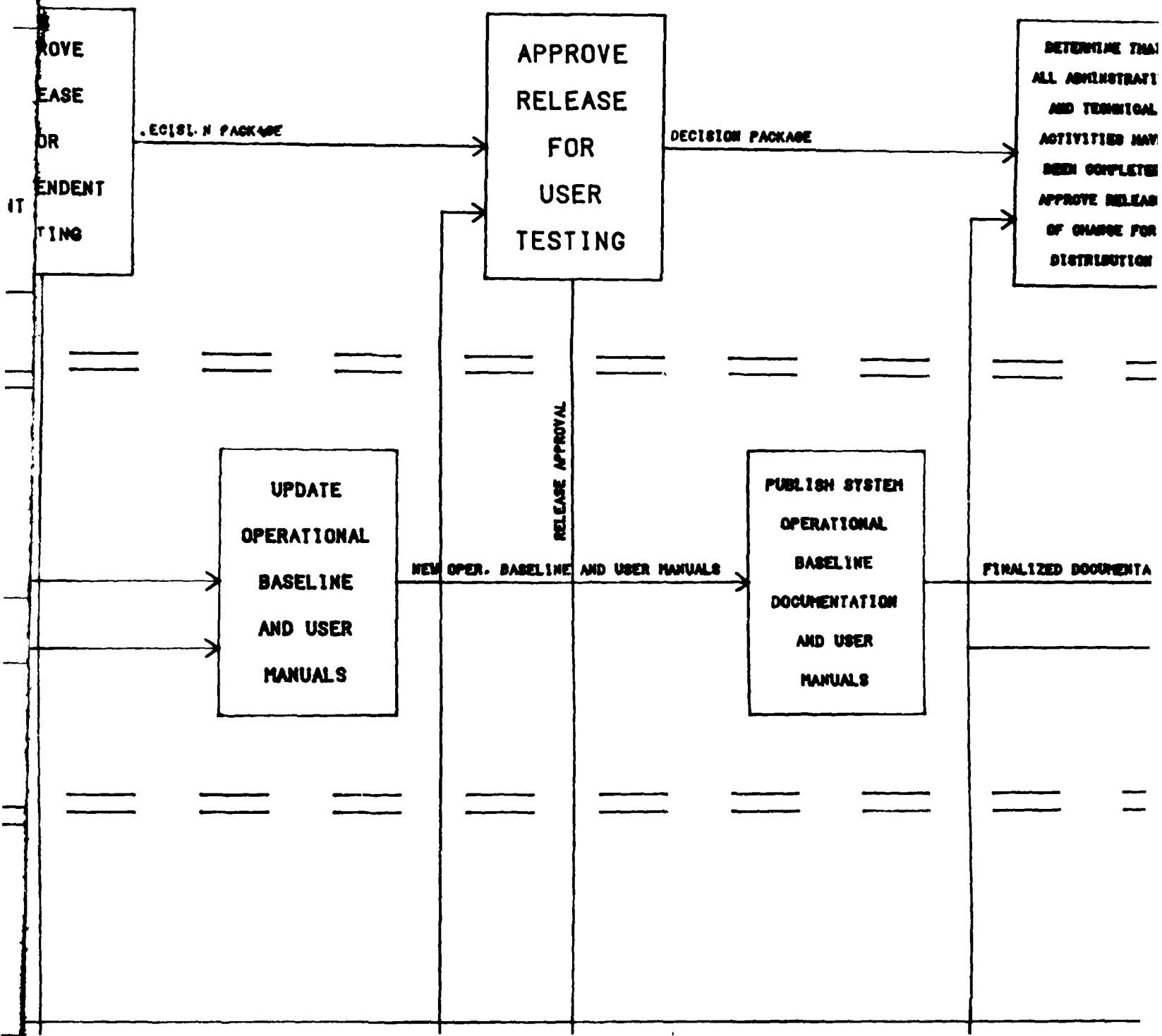
CHANGE PROCES



ISS FUNCTIONAL



MODEL



ONE THAT
ADMINISTRATIVE
COMMittal
SITES HAVE
COMPLETED
THE RELEASE
ARRANGEMENT FOR
DISTRIBUTION

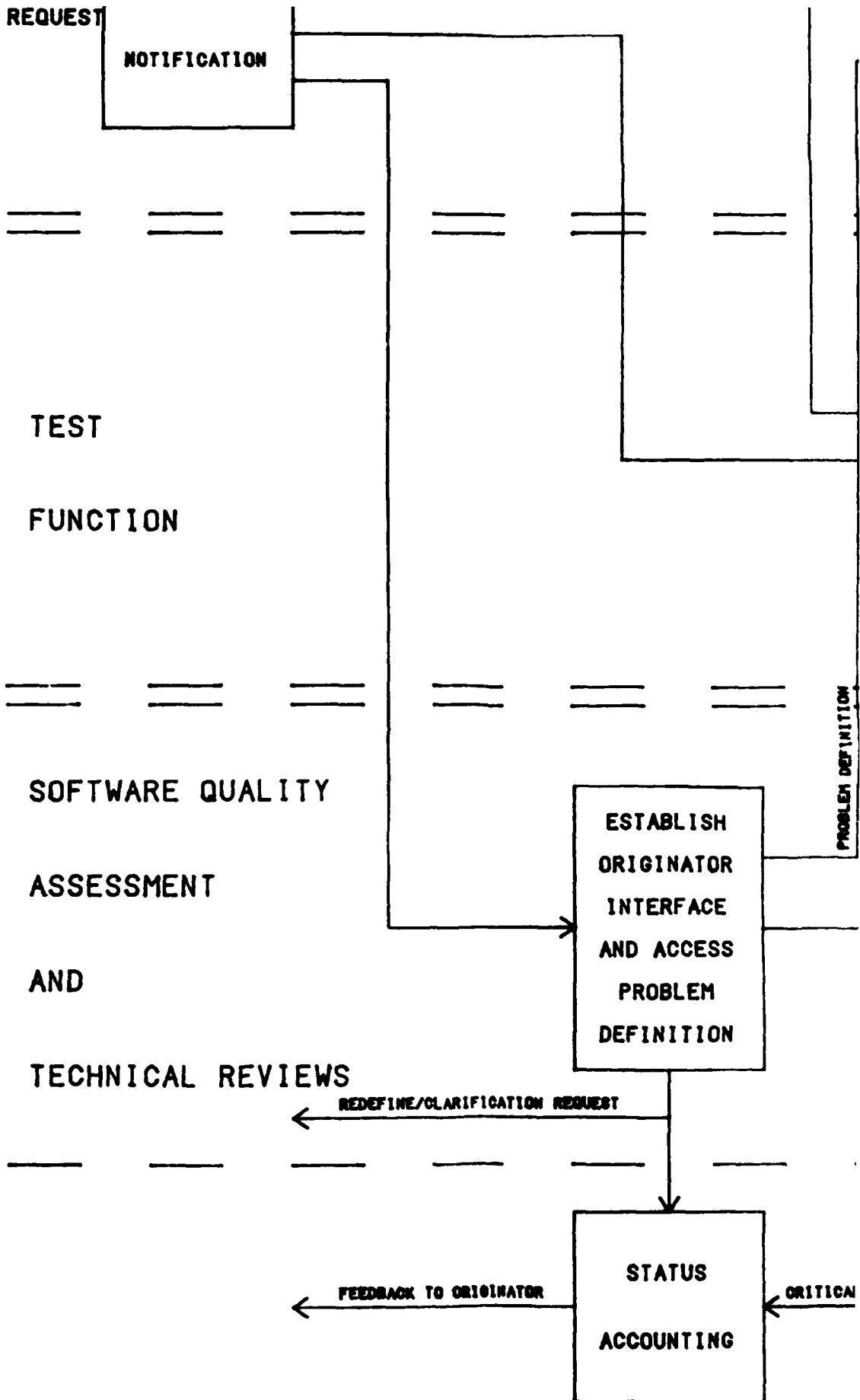
ADMINISTRATION APPROVAL PAGE

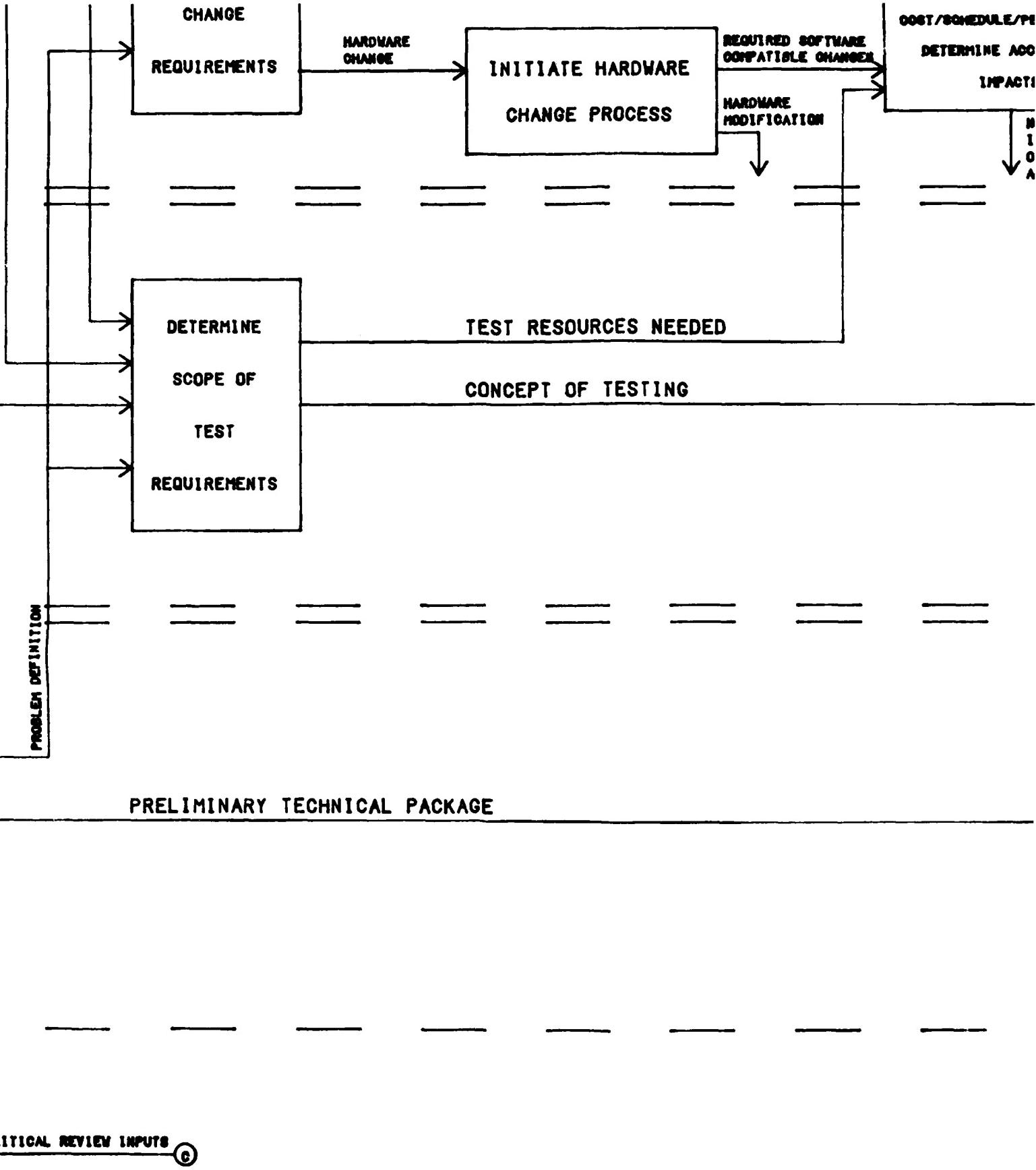
PK
DOCUMENTATION PKG.

DISTRIBUTE
AND
ARCHIVE ALL
DOCUMENTATION

DISTRIBUTE

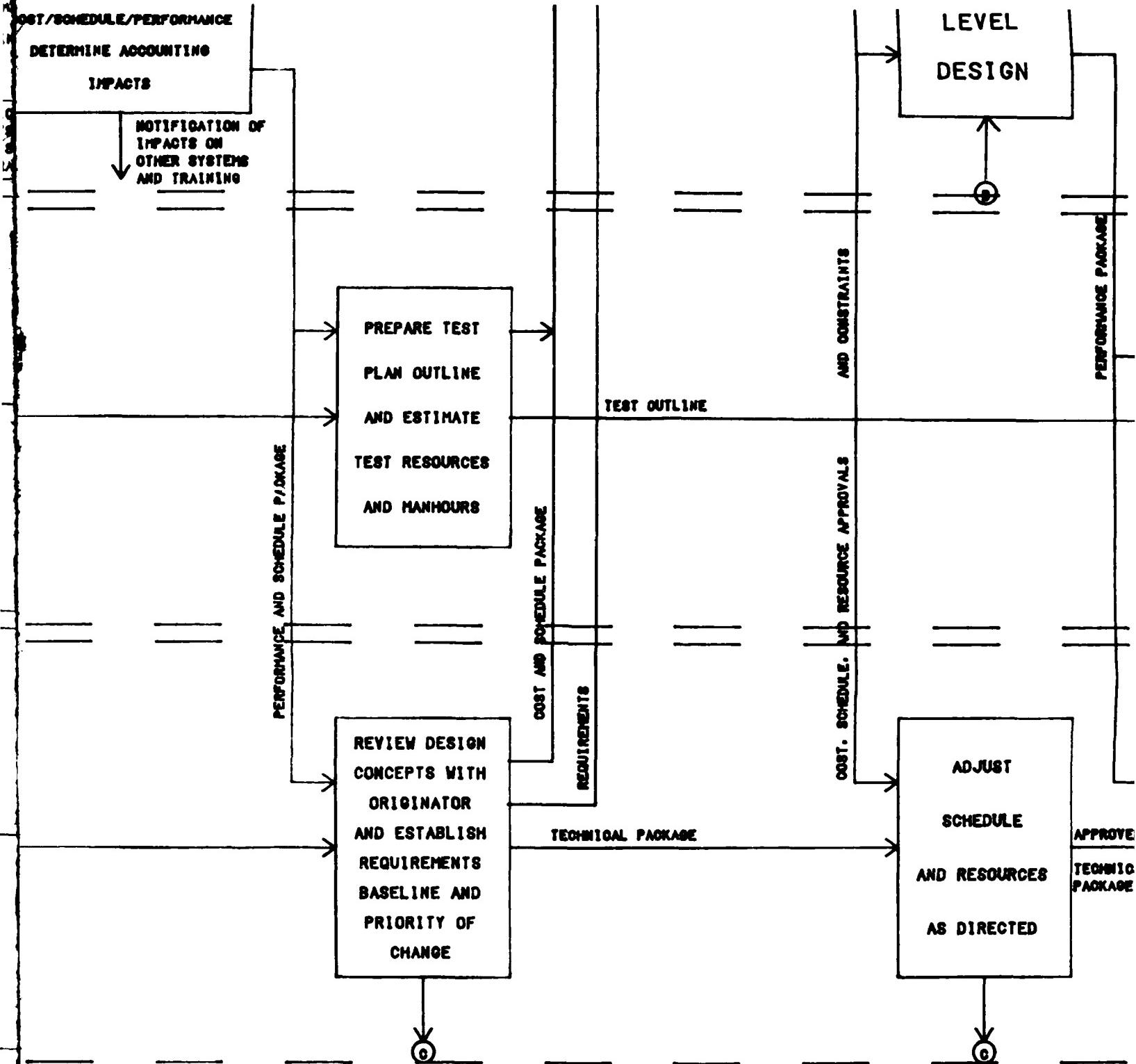
CHANGE

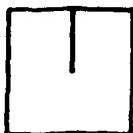
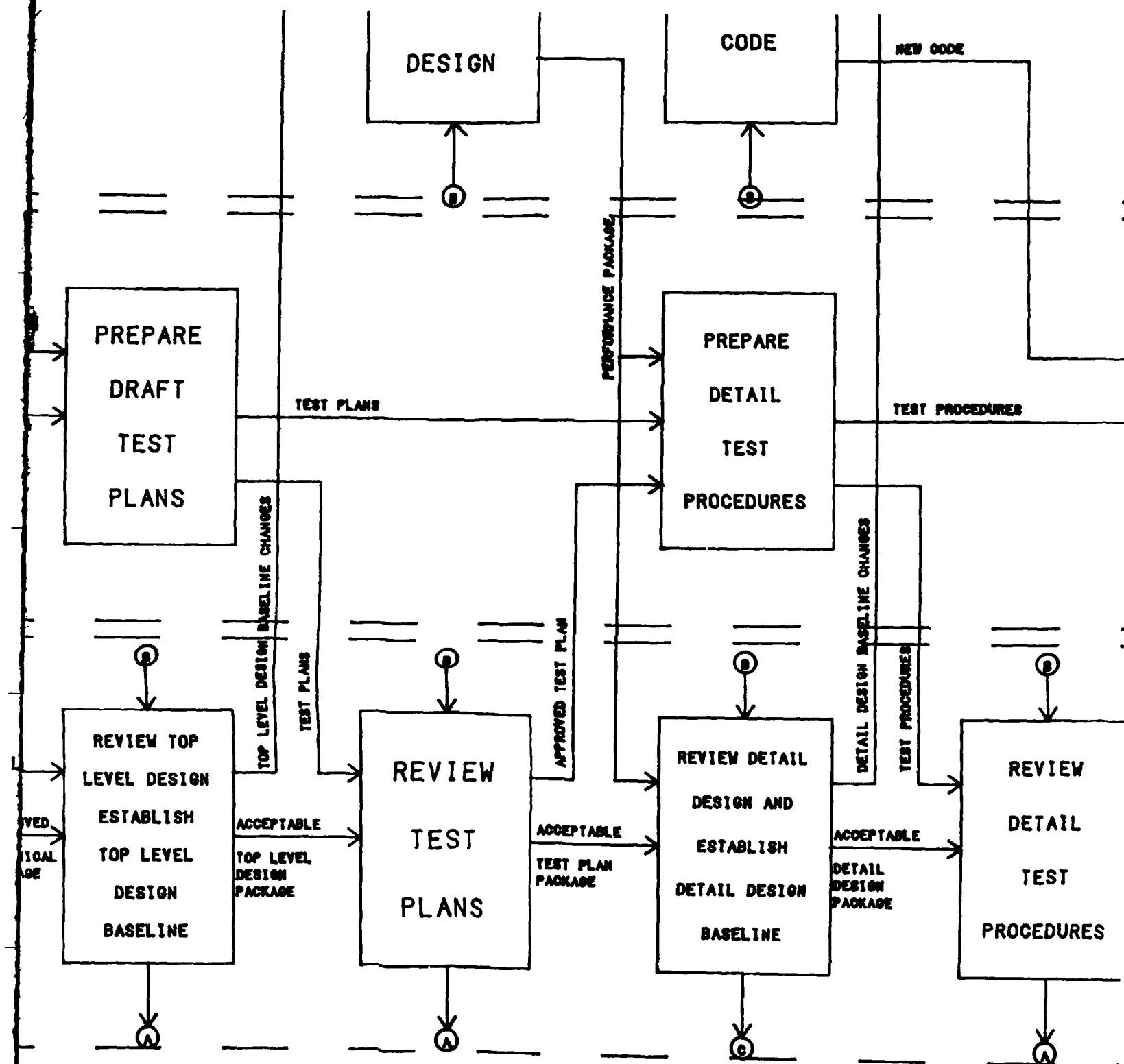


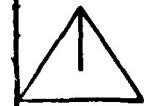
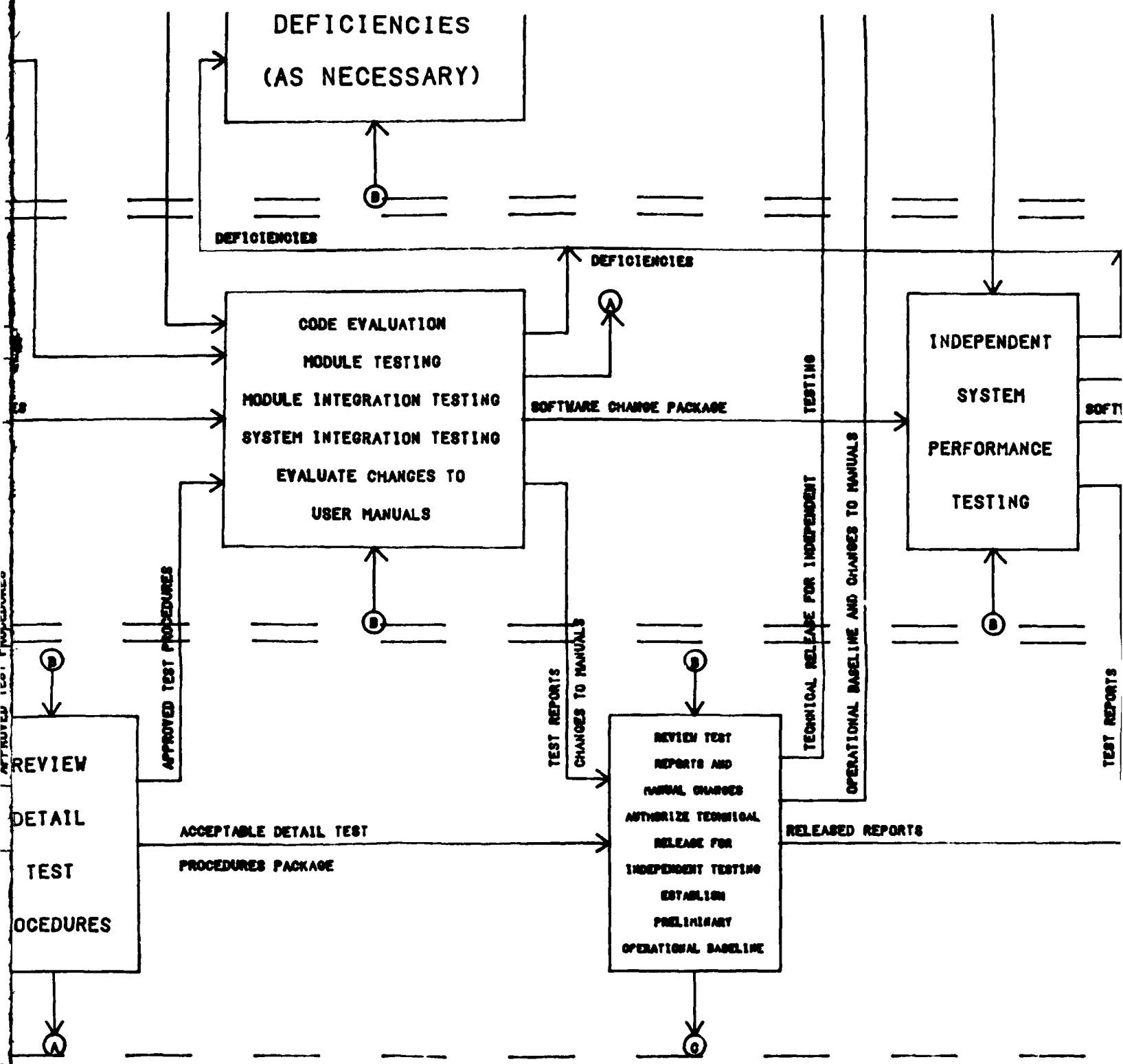


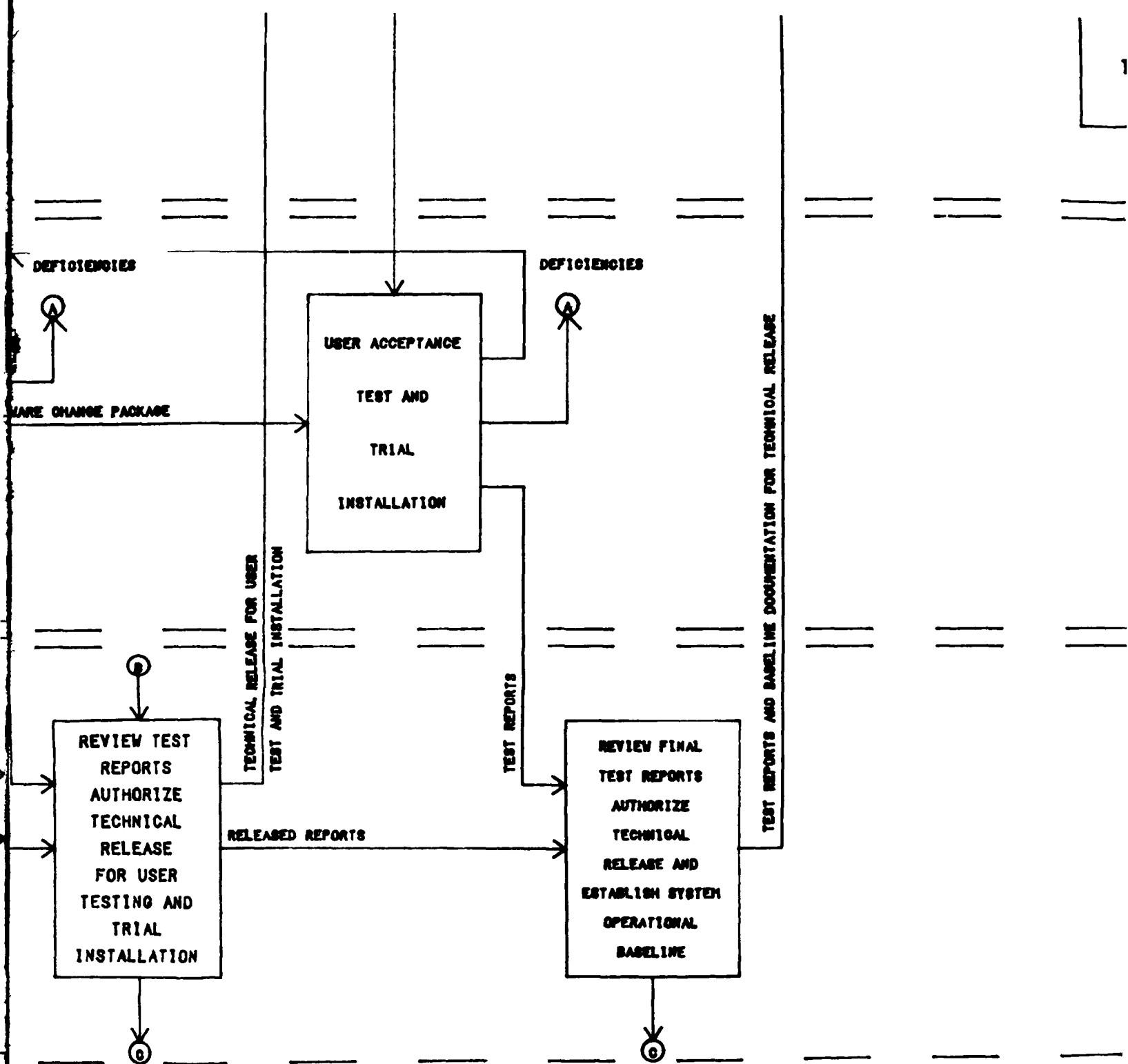
CRITICAL REVIEW INPUTS

(c)









TO USER

=====

=====

=====

**DATE
ILMED
— 8**